

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ**

**Факультет кібербезпеки і програмної інженерії
Кафедра інженерії програмного забезпечення**

**ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри**

“ _____ ” _____ 2023р.

**ДИПЛОМНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)**

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

Тема: “Методологія вдосконалення програм на основі засобів генерації коду
штучним інтелектом”

Виконавець: ст. гр. 221МА Данилов Олексій Геннадійович

Керівник: к.ф.-м.н., доцент, Михайло Вікторович Оленін

Нормоконтролер: Михайло ОЛЕНІН

Київ 2023

**MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
NATIONAL AVIATION UNIVERSITY**

Faculty of cybersecurity and software engineering
Software engineering department

ADMIT TO DEFENSE
Head of Department

“ _____ ” _____ 2023

QUALIFICATION WORK
(EXPLANATORY NOTE)

GRADUATE OF EDUCATIONAL MASTER’S DEGREE

Theme: “Methodology for improving programs based on means of code generation by artificial intelligence”

Performer: Danylov Oleksii Gennadijovich

Standard controller: Doctor of technical sciences, Associate Professor, Mykola Fyodorovych Radishevskiy

Supervisor: Mykhailo OLENIN

Kyiv 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки і програмної інженерії

Кафедра інженерії програмного забезпечення

Освітній ступінь магістр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ 2023 р

ЗАВДАННЯ

на виконання дипломної роботи

Данилова Олексія Геннадійовича

1. Тема дипломної роботи: «Методологія вдосконалення програм на основі засобів генерації коду штучним інтелектом»
затверджена наказом ректора від «29» жовтня 2023 р. № 1994/ст.
2. Термін виконання роботи: з 02.10.2023 р. до 31.12.2023 р.
3. Вихідні дані до роботи : Вдосконалений програмний продукт з використанням VSCode та мов програмування Typescript та Javascript.
4. Зміст пояснювальної записки:
 - 1) Аналіз методології вдосконалення програм на основі генерації коду ШІ,
 - 2) Дослідження технологій впровадження,
 - 3) Архітектура вдосконалення програми,
 - 4) Прототип розробленого програмного забезпечення.
5. Перелік обов'язкового ілюстративного матеріалу (слайдів презентації):
 - 1) Тема, виконавець, керівник;
 - 2) Існуючі методи, аналіз недоліків, постановка завдання;
 - 3) Вимоги до програмного засобу;
 - 4) Структура інструменту, діаграма класів;
 - 5) Інтерфейс програмного засобу;
 - 6) Демонстрація прототипу інструменту

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Відмітка про виконання
1.	Розробка та затвердження графіка роботи. Ознайомлення з постановкою задачі та вивчення літератури. Написання розділів ПЗ, представлення керівнику.	14.10.2023-31.10.2023	
2.	Написання першого та другого розділу. Надання на перший нормоконтроль ПЗ - (обов'язково - титульної, завдання, графіка, реферат, список скорочень, зміст, вступ, список джерел, розділи ПЗ).	15.10.2023-22.10.2023	
3.	Редагування пояснювальної записки, графічного матеріалу.	22.10.2023-01.11.2023	
4.	Проходження першого нормоконтролю.	01.11.2023-14.11.2023	
5.	Написання третього та четвертого розділу дипломної роботи.	14.11.2023-20.11.2023	
6.	Проходження контролю ПЗ на плагіат.	20.11.2023-26.11.2023	
7.	Отримання відгуку керівника. Підготовка презентації та тексту доповіді. Попередній захист.	26.11.2023-27.11.2023	
8.	Проходження нормоконтролю, перепліт пояснювальної записки. Отримання рецензії.	27.11.2023-13.12.2023	
9.	Здати секретарю ДЕК: ПЗ, ГМ, CD-R з електронними версіями ПЗ, ГМ, презентацію, відгук керівника, рецензію, довідку про успішність, 2 папки, 2 конверта.	13.12.2023-19.12.2023	
10.	Захист дипломної роботи перед ЕК.	19.12.2023-31.12.2023	

Дата видачі завдання 02.10.2023р.

Керівник дипломної роботи: _____ Михайло ОЛЕНІН

Завдання прийняв до виконання: _____ Олексій ДАНИЛОВ

NATIONAL AVIATION UNIVERSITY

Faculty cybersecurity and software engineering

Department Software Engineering

Degree of education master

Specialty 121 Software engineering

Education-professional program Software engineering

APPROVED

Head of department

“ ” _____ 2023

Task

on executing the graduation work

Danylov Oleksii Gennadijovich

1. Topic of the graduation work: “Methodology for improving programs based on means of code generation by artificial intelligence”
Approved by the rector's order from «29» october 2023 № 1994/st.
2. Terms of work execution: from 02.10.2023. to 31.12.2023.
3. Source data of the work: improved software product using VS Code and programming languages Typescript and Javascript
4. The content of the explanatory note:
 1. Analysis of methodology for improving programs based on AI code generation
 2. Research of the implementation technologies
 3. Architecture of improved program
 4. Prototype of developed software
5. List of mandatory presentation slides:
 1. Topic, performer, leader.
 2. Existing methods, analysis of shortcomings, setting of the task.
 3. Requirements for the software tool.
 4. Tool structure, class diagram.
 5. Software tool interface.
 6. Demonstration of the tool prototype.

6. Calendar plan-schedule

№	Task	Deadline	Performance note
1.	Familiarization with the statement of the problem and the study of literature Writing 1 section, presentation to the supervisor	14.10.2023- 31.10.2023	
2.	Preprint of section 1 and auxiliary pages (draft) - title, task, schedule, abstract, list of abbreviations, content, introduction, source list. First standard control.	15.10.2023- 22.10.2023	
3.	Writing 2 section, presentation to the supervisor	22.10.2023- 01.11.2023	
4.	Writing 3 section, presentation to the supervisor	01.11.2023- 14.11.2023	
5.	General editing and printing of an explanatory note, graphic material	14.11.2023- 20.11.2023	
6.	Passing standard control	20.11.2023- 26.11.2023	
7.	Development of the text of the report. Creating of graphic material for presentation	26.11.2023- 27.11.2023	
8.	Get feedback from the supervisor, reviews.	27.11.2023- 13.12.2023	
9.	Preparation of materials for transmission to the secretary of the DEC (software, GM, CD-R with electronic copies of software, GM, presentations, supervisors review, review, certificate of progress, 2 folders, 2 envelopes)	13.12.2023- 19.12.2023	
10.	Graduation project presentation	19.12.2023- 31.12.2023	

Date of issue of the assignment 02.10.2023.

Supervisor: _____ Mykhailo OLENIN

Task accepted for execution: _____ Oleksiy DANYLOV

РЕФЕРАТ

Пояснювальна записка до дипломної роботи “Методологія удосконалення програм на основі засобів генерації коду штучним інтелектом”: 93 с., 31 мал., 1 табл., 2 графіки, 7 діаграмм, 10 використаних джерел, 1 додаток.

Об’єкт дослідження – методологія вдосконалення програм на основі засобів генерації коду штучним інтелектом.

Мета роботи – запропонувати методологію вдосконалення програм на основі засобів генерації коду штучним інтелектом та продемонструвати її застосування на практичних прикладах.

Методи дослідження – використання методів реверсивної інженерії програмного забезпечення та методів штучного інтелекту.

Тип розробки: об’єктно-орієнтований підхід.

Апаратне та програмне забезпечення – ПК з операційною системою Windows 11 або Windows 10, середовище для об’єктно-орієнтованого програмування – VS Code. Використання методології штучного інтелекту неможливо без підключення до Інтернету.

Розробка програми виконувалася в середовищі Node.js. Передбачене припущення щодо розробки інструментів – розроблене програмне забезпечення можна покращити шляхом поєднання генерації штучного інтелектуального коду та перевірки вразливостей штучного інтелекту.

Результати роботи – можуть бути використані при розробці програмних засобів або створенні та вдосконаленні програмних модулів існуючої програми, призначених для різних галузей промисловості.

ВДОСКОНАЛЕННЯ ПРОГРАМИ, ГЕНЕРУВАННЯ КОДУ, СТВОРЕННЯ ДОКУМЕНТАЦІЇ, ПОШУК ВРАЗЛИВОСТЕЙ, РЕВЕРСНА ІНЖЕНЕРІЯ, ДОДАВАННЯ ПРОГРАМНИХ МОДУЛІВ.

ABSTRACT

Explanatory note to the thesis "Methodology of program improvement based on means of code generation by artificial intelligence": 93 pp., 31 figures, 1 tables, 2 graphs, 7 diagrams, 10 used sources, 1 appendices.

The object of the research – the methodology of improving programs based on means of code generation by artificial intelligence.

The goal of the work is to propose a methodology for improving programs based on means of code generation by artificial intelligence and to demonstrate its application on practical examples.

Research methods – use of software reverse engineering methods and artificial intelligence methods.

Type of development: object-oriented approach.

Hardware and software – PC with Windows 11 or Windows 10 operating system, an environment for object-oriented programming –VS Code. The use of artificial intelligence methodology is impossible without an Internet connection.

Application development was done in Node.js runtime environment
Intended tool development assumption – developed software can be improved by combining artificial intelligence code generation and AI vulnerability testing.

The results of the work can be used in the development of software tools or the creation and improvement of software modules of an existing program intended for various industries.

**SOFTWARE IMPROVEMENT, CODE GENERATION,
DOCUMENTATION CREATION, VULNERABILITY FINDING, REVERSE
ENGINEERING, ADDING SOFTWARE MODULES.**

TABLE OF CONTENT

LIST ACRONYMS AND ABBREVIATIONS	
INTRODUCTION.....	
CHAPTER I.....	
RESEARCH OF THE METHODOLOGY FOR IMPROVING PROGRAMS CODE	
1.1. Background	
1.2. Research Objectives.....	
1.3. Methodology	
1.3.1. Overview of AI Techniques	
1.3.2. Technologie Integration.....	
1.4. Practical studies of use.	
1.4.1. Overview	
1.4.2. Describing of existing project.....	
1.4.3. ChatGPT request generation	
1.4.4. Github Copilot code generation	
1.4.5. IntelliSense reference autocomplition	
1.4.6. Code formatter manual preferences and work	
1.4.7. Bug fixing	
1.5. Signs of methodology	
1.6. Analysis of literary sources and existing analogues of AI code generators and AI code analyzers.....	
1.6.1. Analysis of analog programs	
1.6.2. Found software analogs to artificial intelligence Github Copilot.	
1.6.3. Comparison of existing analogue programs	
1.7. Source understanding.....	
1.8. Relevance of the improving programs by ai code generation.....	
1.8.1. Benefits and Challenges of AI-Driven code generation	
1.8.2. Future Directions	
1.8.3. Formulation of the problem of using these technologies and possible solutions	
1.9. Challenges.....	
Conclusion.....	
CHAPTER 2	
REQUIREMENTS FOR THE IMPROVEMENT OF THE CODE GENERATION SYSTEM.....	
2.1. Requirements for the designing of software.....	
2.2. Description of the diagram	
2.3. Functional requirements	
2.3.1. Capabilities for backend part of web app applications	
2.3.2. Capabilities for frontend part of web app applications.....	
2.4. Non-functional project requirements.....	
2.4.1. Requirements for backend part of web app applications.....	
2.4.2. Requirements for frontend part of web app applications	
2.5. Detailed Description	
2.5.1. Capabilities for backend part of web app applications	
2.5.2. Capabilities for frontend part of web app applications:.....	
2.5.3. Requirements for backend part of web app applications.....	

2.5.4. Requirements for frontend part of web app applications	
Conclusion.....	
CHAPTER 3	
PROPOSED ARCHITECTURE OF THE SOFTWARE APPLICATION.....	
3.1. Software architecture	
3.2. Diagram of the deployment of the link shortening web application	
3.3. Component diagram of the link shortening web application.....	
3.4. Class diagram.....	
3.5. Construction of connections of physical equipment	
Conclusion	
CHAPTER 4	
PROTOTYPE IMPLEMENTATION USING THE CODE GENERATION METHODOLOGY	
4.1. Development of a prototype of a web service for shortening links using the methodology of using artificial intelligence	
4.2. Prototype of the developed system	
4.3. User interface development.	
4.4. Results of web link reduction system testing.	
Conclusion	
CONCLUSIONS	
LIST OF REFERENCES	
APPENDIX A.	

LIST ACRONYMS AND ABBREVIATIONS

IDE – Integrated development environment;

JS – JavaScript programming language;

TS – Typescript object oriented programming language;

NE (Node.js) – Environment system for javascript and typescript;

EX (Express.js) – Framework that working with Node.js and webapps

SQ (Sequelize) – Typescript and Node.js ORM for different databases;

RD (Redis) – in memory data structure store, used as database (cache);

DB – database;

OOP – object oriented programming;

OS – operational system;

DBMS – database management systems;

DAO (Data Access Object) – a module that consists of one or more files in the program, which is responsible for communication with the database;

GPT (Chat GPT) – Multifunctional AI that works in text solution that can create some ideas, problem solves, code understanding;

CF (Code Formatter) – API or Integrated software that works with endup code to convert it to more understandable and readable for developer view;

IS (IntelliSense) – Autocomplete technology that provide a developer mostly view interface of different variants of references autocompletion that will automatically generate code;

GC (Github Copilot) – AI tool that can generate module/function code and give variants of realization of some task, as well as bug fixing;

CW (Amazon CodeWhisperer)– AI tool for searching security vulnerabilities and code stable analyzer;

CM (Codeium) – Free AI tool alternative to Copilot;

TE (Tabnine) – Free AI tool alternative to Copilot;

FP (Faux Pilot) – Open source self hosted copy of old version of Copilot;

AI – Artificial intelligence;

INTRODUCTION

The methodology of designing software modules using artificial intelligence plays a crucial role in understanding and analyzing existing software systems. This paper examines the integration of artificial intelligence (AI) techniques, including IntelliSense, Code Formatter, ChatGPT, and GitHub Copilot and similar programs to GitHub Copilot for code generation of software systems.

The goal is to use AI capabilities to improve various software modules, including code understanding, refactoring, and creating step-by-step documentation for performing specific software improvement actions. We review each AI method, discuss their potential contribution to improved part design, and present a framework for their integration. In addition, we discuss the benefits, challenges, and future directions for AI-driven improvement and exploration of critical security challenges code generation model.

The relevance of the creation of program modules by artificial intelligence as a scientific topic at the moment is difficult to overestimate, since in our time there is a rapid development of artificial intelligence, its use can add an increase in speed and create opportunities for the integration of AI in projects to obtain better software. The use of artificial intelligence does not mean a complete rejection of developers, they still need to develop and have the necessary level of knowledge to create software, and the involvement of artificial intelligence in the development process can contribute to positive changes in the design process, since artificial intelligence cannot create a quality product that consists of from many modules itself, but only to copy already existing software analogs and functions that this analog has.

The methodology of creating software modules is strictly connected to meaning of reverse engineering since this methodology involves viewing an existing software, therefore, it is appropriate to use this concept as compatible with this type of work

CHAPTER I

RESEARCH OF THE METHODOLOGY FOR IMPROVING PROGRAMS CODE

1.1. Background

AI code generation involves analyzing and understanding existing software systems to gain valuable insights, improve maintainability, and support future development efforts. Traditional approaches to code generation rely on manual analysis, which can be time-consuming and error-prone in inferences to existing code. The problem statement in this area is that the manual nature of project support and improvement creates challenges in terms of efficiency, accuracy and scalability. Therefore, there is a growing need to use AI methods to automate and improve various aspects of the design process.

1.2. Research Objectives

The main objectives of AI integration:

- To explore the integration of IntelliSense, Code Formatter, ChatGPT, and GitHub Copilot for AI-driven code generation.
- To evaluate the effectiveness of the combined AI approach in improving code comprehension, refactoring, and documentation generation during developing software module.
- To identify the benefits, challenges, and ethical considerations associated with AI-driven module development.
- To propose future directions and recommendations for advancing AI techniques in time of development.

1.3. Methodology

1.3.1. Overview of AI Techniques

This section provides a brief explanation of the AI techniques used in the integrated approach: IntelliSense, Code Formatter, ChatGPT, and GitHub Copilot. It describes their functionalities.

IntelliSense: Improving Code Comprehension IntelliSense goes beyond traditional code completion by leveraging AI algorithms to provide context-aware suggestions and documentation (Fig. 1.1.). In the context of developing, IntelliSense can assist in understanding complex code structures by offering relevant suggestions based on the current code context. For example, it can help identify the parameters and return types of functions, provide documentation for libraries or APIs, and suggest possible code fixes to improve the comprehension of legacy codebases.



Fig. 1.1. “IntelliSense software logo”

IntelliSense is currently supported in Visual Studio for languages such as Java, C++, C#, Javascript, Typescript J#, Visual Basic, Visual FoxPro, XML, HTML, XSLT, and others.

Code Formatter: Automating Refactoring Code Formatter utilizes AI algorithms to automatically analyze and refactor existing code, making it more readable, maintainable, and aligned with coding best practices (Fig. 1.2.). In the context of developing, Code Formatter can aid in refactoring legacy codebases by identifying and applying improvements to the code structure, naming conventions, indentation, and other formatting aspects. This automated refactoring not only enhances code readability but also facilitates the identification of code patterns and structures during the developing process.



Fig. 1.2. "Example of code formatter Prettier"

We have different code formatters, they all have some benefits and disadvantages so we will not not draw attention to their different options and methods of adjustment. In addition, their installation and connection have the same structure, in which it is easy to install the extension and create a file with the setting.

ChatGPT: Assisting in Documentation Generation ChatGPT, a powerful language model trained on a vast corpus of text data, can play a significant role in developing by assisting in the generation of documentation (Fig. 1.3.). During the development process, ChatGPT can be used to automatically generate explanations, comments, and summaries that describe the functionality, purpose, and relationships between code components. This documentation can serve as a valuable resource for understanding the system's architecture, dependencies, and behavior, particularly in cases where comprehensive documentation may be lacking.



Fig. 1.3. "ChatGPT software logo"

ChatGPT can be integrated with Visual Studio, VS Code, JetBrains IDE, IntelliJ IDEA, etc.

GitHub Copilot: Intelligent Code Generation GitHub Copilot leverages AI models trained on vast code repositories to suggest code snippets, functions, and even entire code blocks based on the observed context (Fig. 1.4.). In development, GitHub Copilot can expedite the understanding of complex code structures by generating relevant code segments that align with the observed behavior or patterns within the software system. These intelligent code suggestions can assist in identifying crucial functionality, understanding system interactions, and accelerating the main process.



Fig. 1.4. "Github Copilot software logo"

GitHub Copilot is trained on all languages that appear in public repositories. For each language, the quality of suggestions you receive may depend on the volume and diversity of training data for that language. For example, JavaScript is well-represented in public repositories and is one of GitHub Copilot's best supported languages. Languages with less representation in public repositories may produce fewer or less robust suggestions.

Codeium, Tabnine and FauxPilot work in the same way, the main difference between them is that they are free to use, the Copilot unlucky have a paid subscription.

1.3.2. Technologie Integration

The main structure of AI work will be the code generation by the request of our Developer (Fig. 1.5.). Our engineer creates an idea of the software view, next by existing problems he needs to solve, write to the ChatGPT what he wants to fix or create and take some information on how to do that. By the answer he can create code by writing comments in code lines of projects we improving and generate some working code.

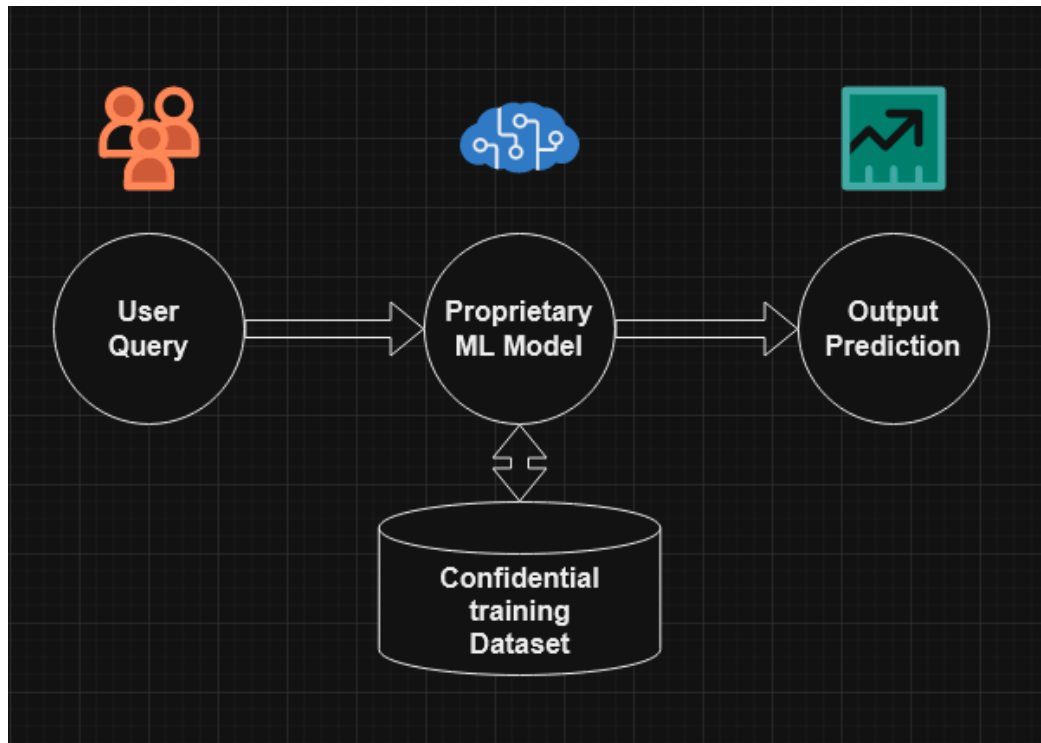


Fig. 1.5. “Artificial intelligence work structure”

We will use the creation of code by ChatGPT named ChatGPT as well as Github Copilot to take the best variants of AI code answers.

If the answer from ChatGPT and Github Copilot don't work, developers at the same time can write to Copilot to fix the problems with code or regenerate the request.

To integrate our technologies we need to perform several steps.

We will use two types of Chat GPT:

1. Browser version for code ideas and solutions.
2. CodeGPT to create some code.

Installation steps:

1. Install the IDE and open the already created project.
2. On the top left of the panel, select the "extensions" tab.
3. Enter "chat gpt" in the search panel and open the tab.
4. Press the installation button.
5. Make the same steps for "Github Copilot" та "Prettier".

Usage steps Chat GPT(Code GPT):

1. After installing the extension, you need to create a Chat GPT Account on the OpenAI website.
2. Next you need to open your profile in OpenAI and create your API Key.
3. Open command palette in Visual Studios and choose "CodeGPT set api key".
4. Now you can choose.

Usage steps Github Copilot:

1. After installing the extension, you can already open any file of your project and start writing some requests.
2. In writing time you will see auto completion of Copilot because it start working and gives you some output even on the start. You can push the "Tab" button to autocomplete requests.
3. To start generating code, push "Enter" button.

Other AIs have similar steps to install.

1.4. Practical studies of use.

1.4.1. Overview

As we see, the listed opportunities provided by tools for project improvement inherit the logic of reverse engineering, since this methodology involves viewing an existing software solution, its research and the creation of additional software modules.

In this section we will see examples of usage of these technologies with each other. As well as describing the model of reverse engineering. Here we can already simply understand why it is so progressive and helpful in developing models.

Let's look at different practical usage of our technologies and artificial intelligence. We will start by each stage of development:

1.4.2. Describing of existing project

Firstly, we defined the existing project points which need to be added. In our case we have already created a project for customers but we need to improve this product and check the vulnerability in it. This project makes any link the user sends to it shorter. According to experimental project name “Link shortening API” which has already been created we have next structure of the project

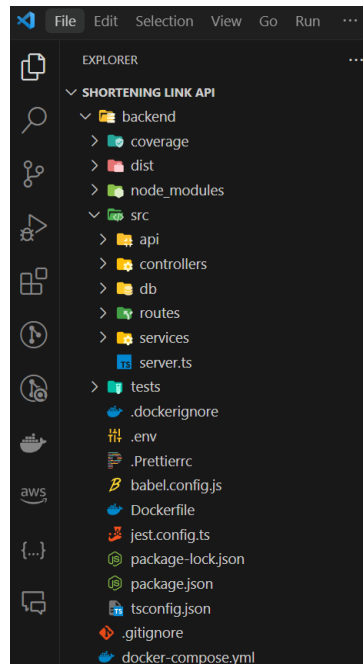


Fig. 1.6. “ChatGPT documentation request”

Main technology stack of the project:

- Environment Node.js;
- Programming language TS and JS;
- Back-end connection to DB is Sequelize;
- Back-end database MySQL;
- “Link” model for DB;
- Fast requests provided by Redis db as application (cache);
- Containerization for starting project with Docker;
- Env file for collection of config credentials;
- Basic CRUD architecture;
- API folders and files to connect link shortening feature;

According to that technology stack our task is to develop several improvements in a project:

(Back end part)

- Change database type due to customer's wishes;

(Front end part)

- Develop the front end part of application to make full-stack app;

(Revision part)

- Audit the project as a whole for security issues and check architecture of app for further scale the project;

Based on the received data, it will be advisable to proceed the operations with artificial intelligence.

1.4.3. ChatGPT request generation

“Change the db type of backend project part by Sequelize from MySQL to PostgreSQL” (Fig. 1.7.) and code input (addition 2.1.).

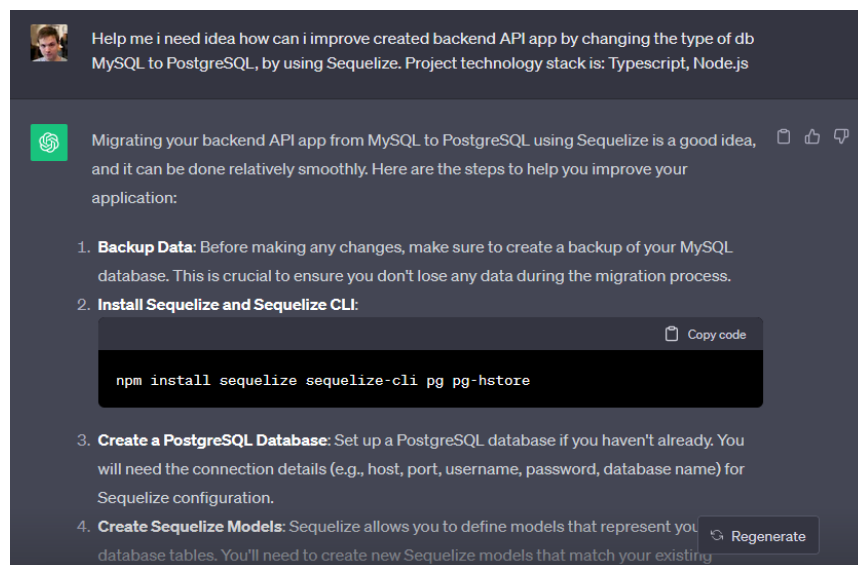


Fig. 1.7. “ChatGPT documentation first part of response”

Now we can see that artificial intelligence is starting to write an algorithm for the developer with steps that must be performed to get the desired result. First of all, in his opinion, it is necessary to copy the data of the old database so as not to lose important information. Next, the development stages begin. He offers to install the necessary components that we already have, so we will skip this step. Next, it is necessary to install the database we need and change the corresponding

model for the database, because before that we had a model for the another database.

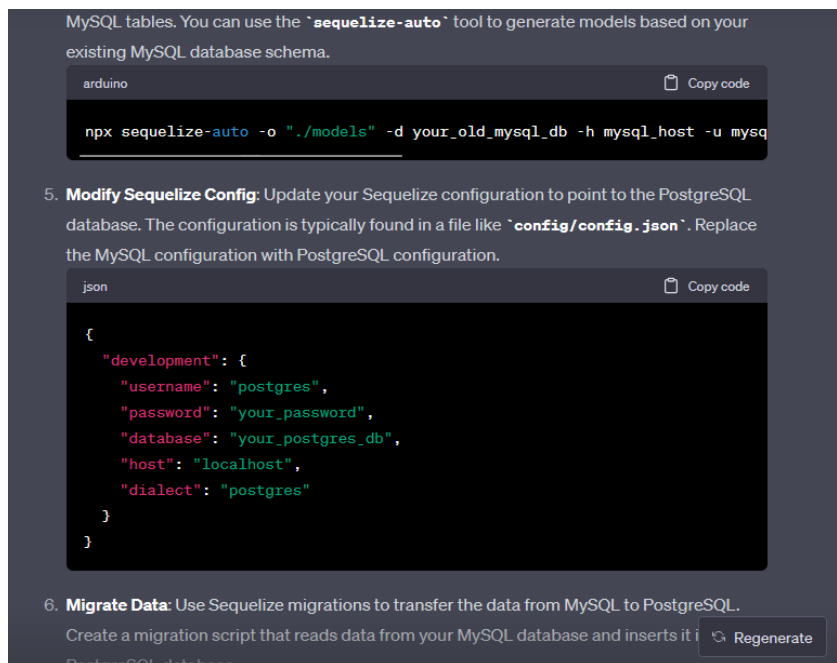


Fig. 1.8. "ChatGPT documentation second response"

Now he provided information in which specific files need to change code in order for the application to create a connection with the database we need.

It is necessary to generate the model with the following command suggested by artificial intelligence, correct it a little, and also change the configuration file, which contains confidential data necessary for the work of the database.



Fig. 1.9. “ChatGPT documentation third response”

Next, he offers to check different aspects of the program for possible problems that may appear after changing the data. Let's take these requirements into account and check the entire application for relevant problems.

- 1) The request and response about developing the front end part of the application to make a full-stack app (Fig. 1.10.)

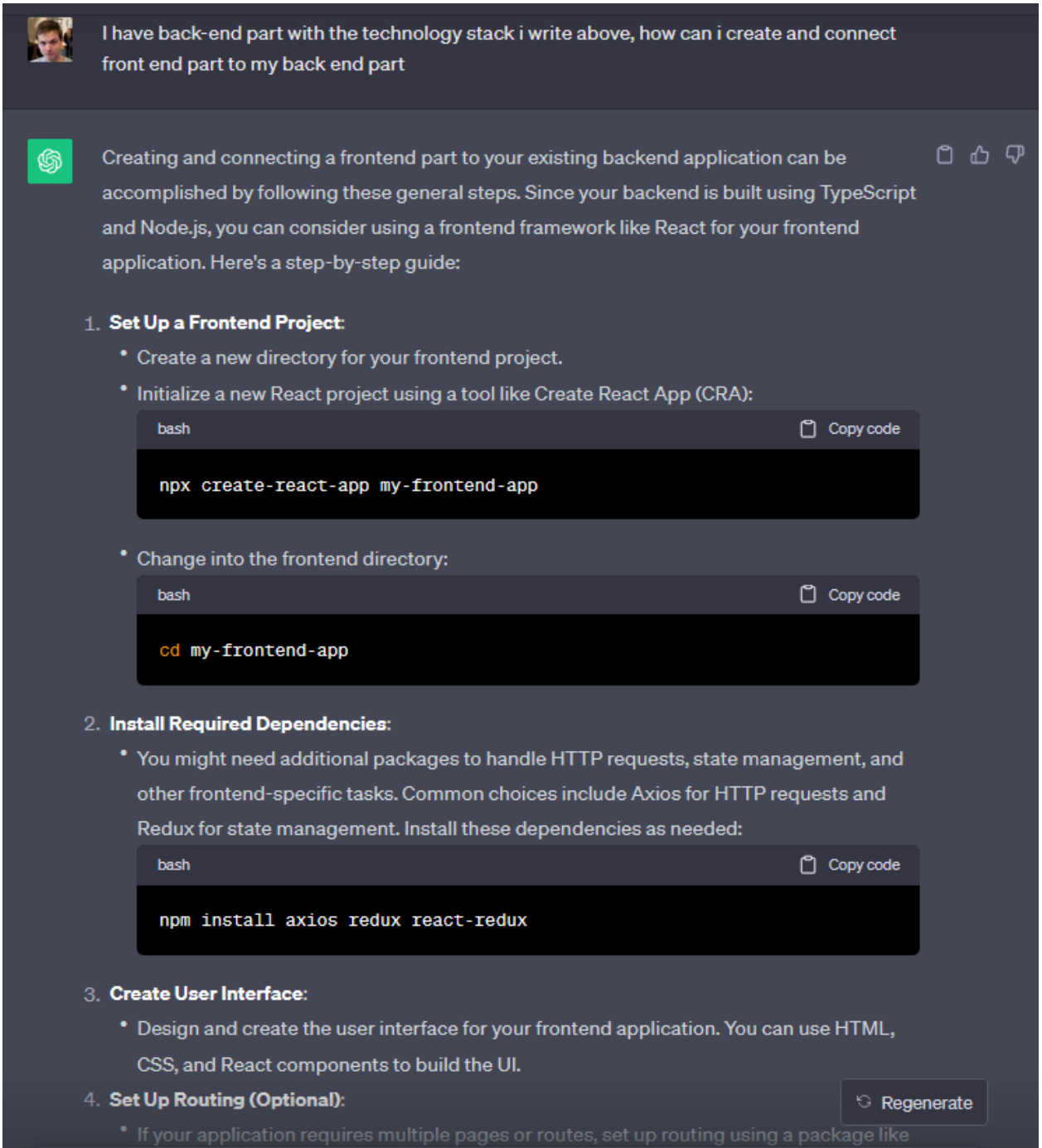


Fig. 1.10. “ChatGPT documentation request”

In the image above, artificial intelligence offers an option on how to create a front-end part of the application. As usual, first of all, creating a file, then installing all the necessary software modules, libraries and frameworks that will be needed during development. After that, with the help of the command in terminal, a starting front-end application is created, and the developer begins to modify it and supplement it with the necessary functions

4. Set Up Routing (Optional):

- If your application requires multiple pages or routes, set up routing using a package like `react-router-dom`.

```
bash Copy code  
  
npm install react-router-dom
```

• Define the routes and create components for each route.

5. HTTP Requests to the Backend:

- Use Axios or another HTTP client to make requests to your backend API. For example, you can make GET, POST, PUT, or DELETE requests to interact with your backend endpoints.

```
javascript Copy code  
  
import axios from 'axios';  
  
const backendUrl = 'http://localhost:your_backend_port';  
  
// Example GET request  
axios.get(`${backendUrl}/api/some-endpoint`)  
  .then(response => {  
    // Handle the response data here  
  })  
  .catch(error => {  
    // Handle errors here  
  });
```

6. State Management (Optional):

- If your application requires state management, configure Redux or another state management library. Create reducers, actions, and connect components to the store to manage application state.

7. Connect to Backend API:

- Use the same API endpoints and routes that you defined in your backend application. Ensure that the frontend and backend are running on compatible ports and domains.

Regenerate

Fig. 1.11. “ChatGPT code change request”

Here it offers manual installation and in-app routing setup. Since the framework with the help of which the front-end part of the application is developed is created as a single-page application (an application in which actions take place on one link that does not change, during these actions, program modules are dynamically interchanged and everything is displayed without changing the link of the web page), it is necessary to add additional technology that will react to the user's actions and replace the web page link so that the user can conveniently understand that he is already on the another page. Next, he creates an example of

connecting requests to the back-end part of the application to send the necessary data

2) Request and response about vulnerability in our project with AWS CodeWhisperer

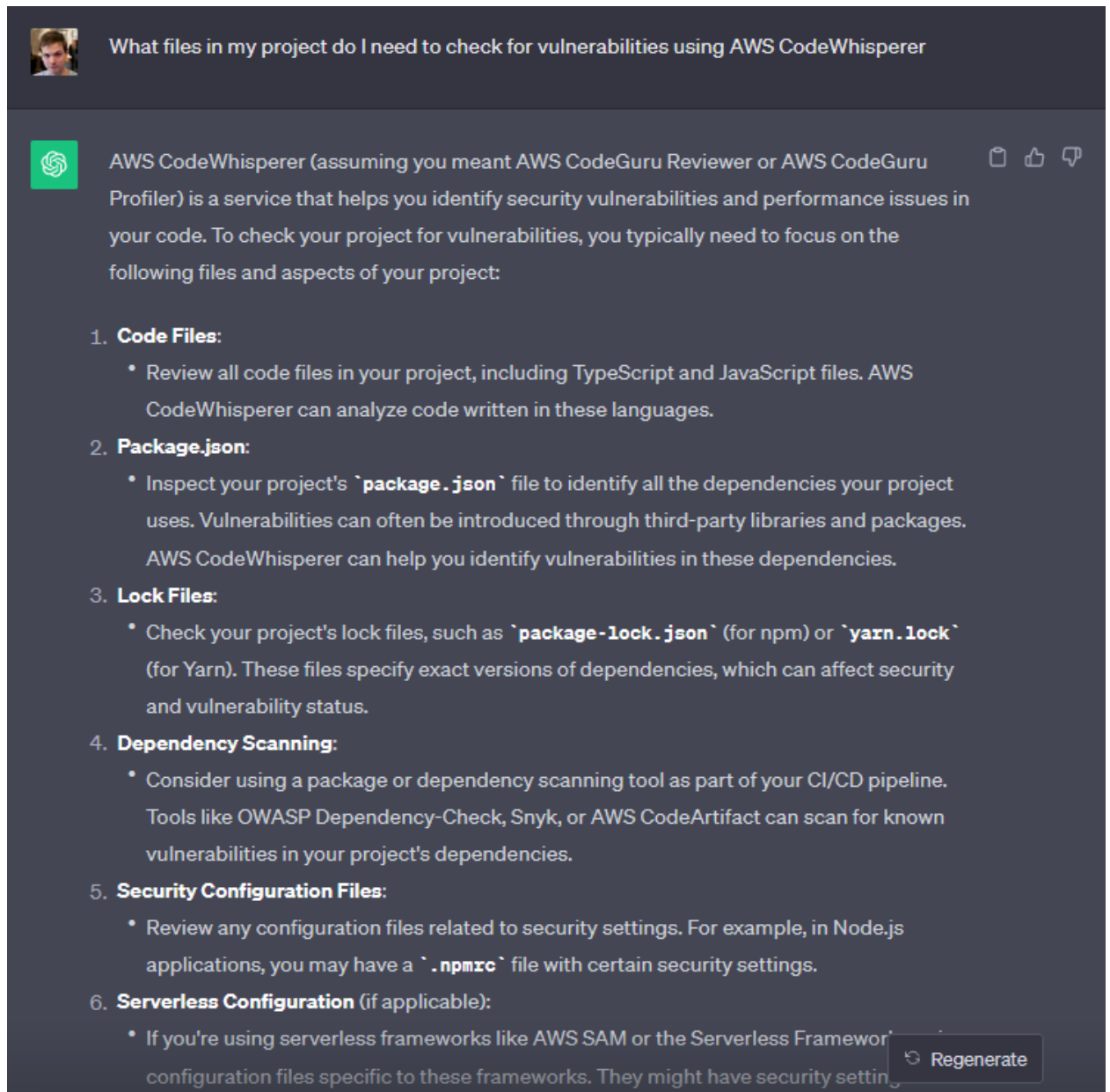


Fig. 1.12. "ChatGPT documentation first response"

In the image above, the answer from the artificial intelligence makes it possible to understand how the check for possible problems with the security of the application works, and how it is necessary to connect the artificial intelligence CodeWhisperer so that it can check the project for possible problems.

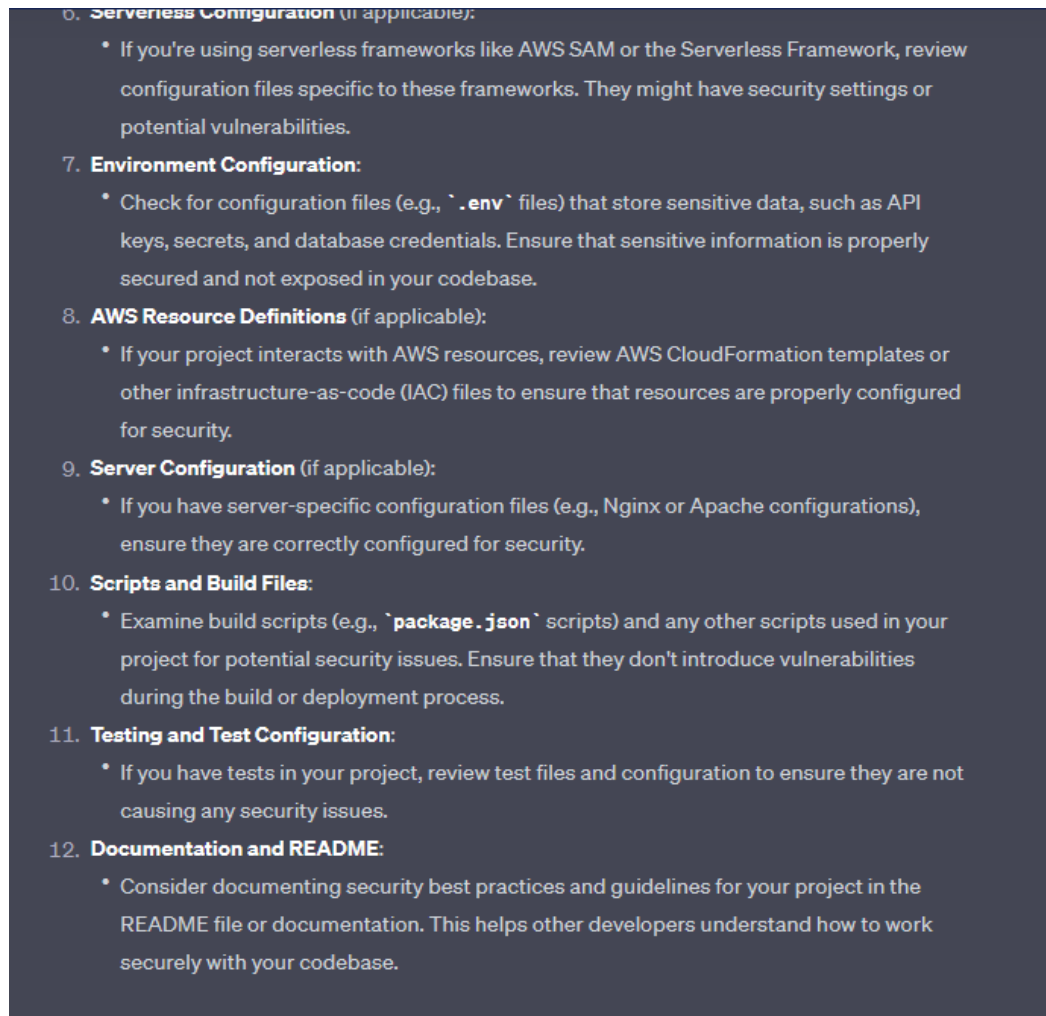


Fig. 1.13. “ChatGPT documentation second response”

Continuation of the previous answer from AI, here he describes all additional necessary information, the developer can quickly read it and easily start performing this work.

Indeed, we can use that documentation for further developing.

1.4.4. Github Copilot code generation

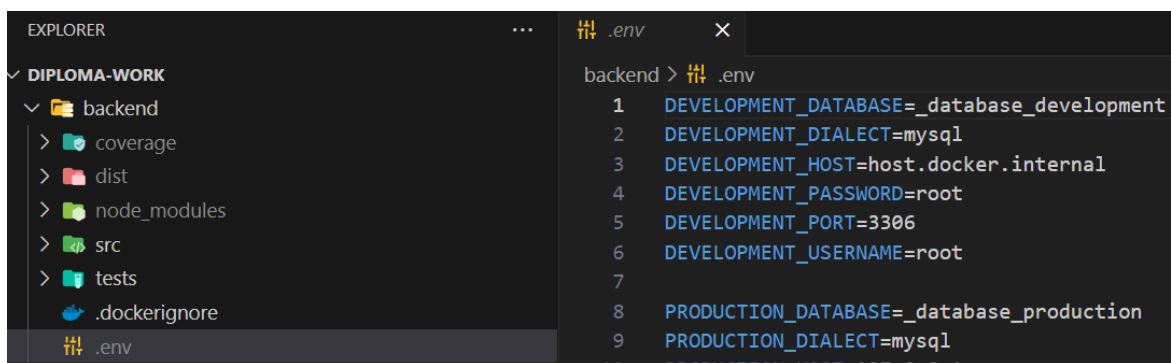


Fig. 1.14. “Project .env configuration file”

The developer writes a request to the AI in which he asks to offer options for changing the database in an already existing application and waits for a response with examples that he can use in the future.

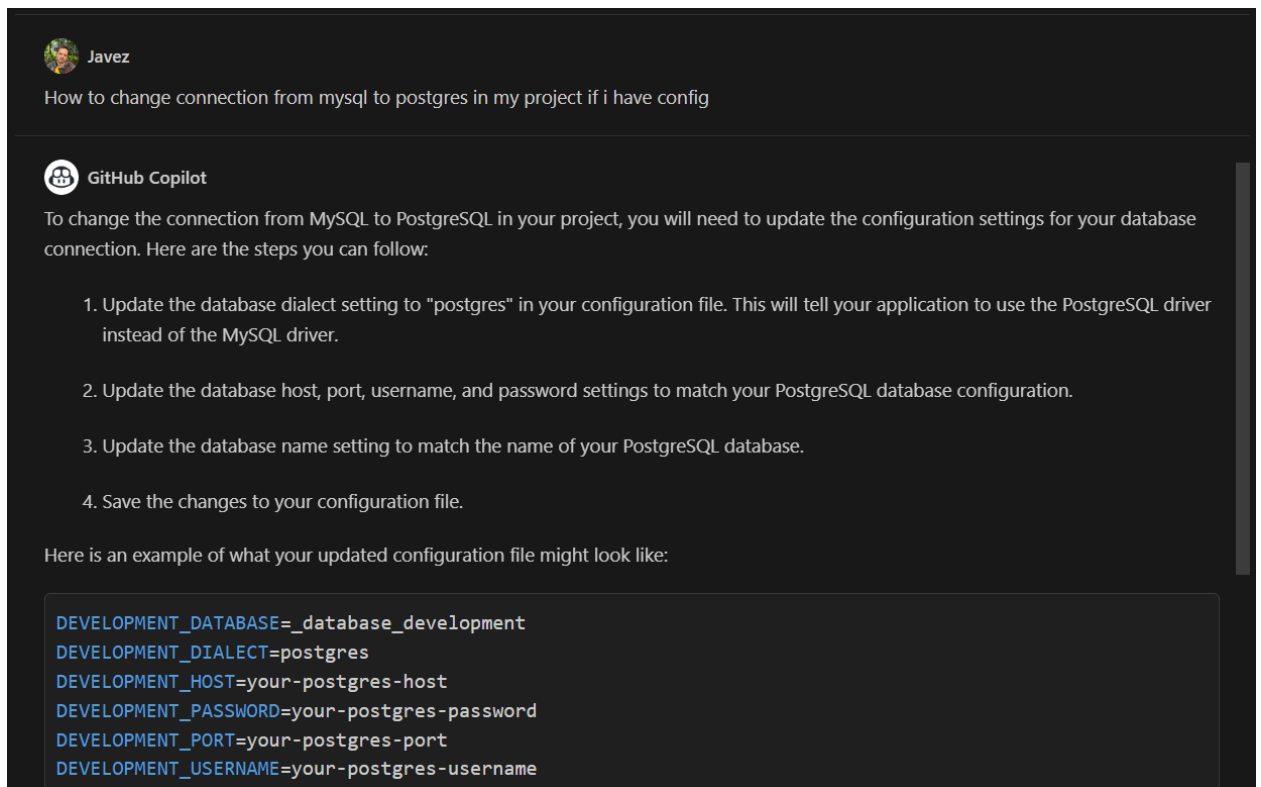


Fig. 1.15. "DB code generation response"

- 3) Now we take the output with exactly the comments we need for our second AI Github Copilot. (Removing some comments that do not belong to our file and need to be extracted to another file).
- 4) Using Copilot changing our comments to code with additional code we need to use.
- 5) Additional instructions:
 - Create auth file in frontend folder.
 - Add the main react function to this file.
 - Change some html code.
- 6) Image below is the code we already have.

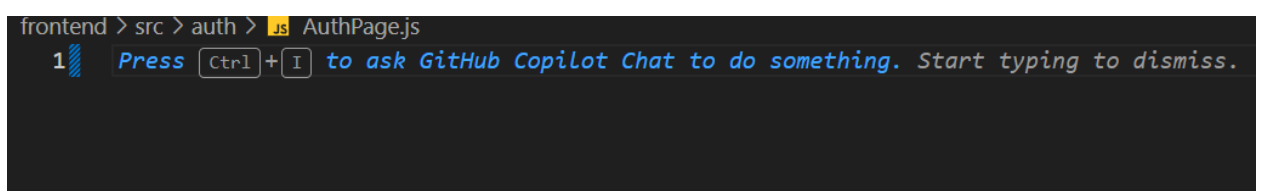


Fig. 1.16. “Github copilot code generation”

- 7) Now by existing comments we start to use Github Copilot. Just start to write some additional text in comments and the AI will understand what you want to start creating the code (Fig. 1.17.).

```
frontend > src > components > auth > JS auth.js
...
1 //create auth react function with several methods for authentication
2 //code here...
3
4 export default function AuthPage() {
}
}
```

Fig. 1.17. “Commented tasks for code generation”

- 8) By each line we see how AI gives us examples and by pushing “Enter it starts code generation (Fig. 1.18.).

```
frontend > src > components > auth > JS auth.js > AuthPage
...
1 //create auth react function with several methods for authentication
2 //code here...
3
Codeium: Refactor | Explain | Generate JSDoc
4 export default function AuthPage() {
5   return (
    <div>
      <h1>Auth Page</h1>
    </div>
  );
6 }
```

Fig. 1.18. “Code generation example”

In the image above, the developer has written comments that artificial intelligence also picks up for the next correct answer. In response to the request, the artificial intelligence immediately picks up the comments and gives the assumed code that is needed, it is highlighted in gray. After pressing the Tab button, the generated code is added to the file and the AI immediately offers the following code or actions.

- 9) The same steps we continue to do with the other comments into another file (Fig. 1.19.)

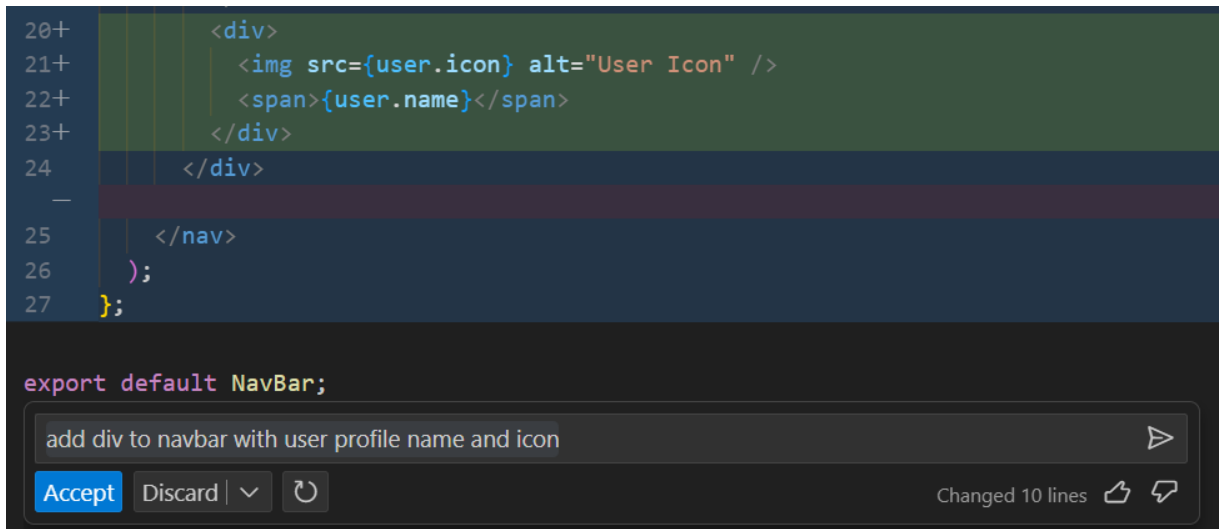


Fig. 1.19. “Code generation with panel request”

Another option for interaction with AI is the creation of a direct request and in the file. To do this, you need to press Control + I, then you need to enter a request with all additional descriptions of the task. It is also necessary to take into account the fact that the generated code will appear in the line previously selected by the mouse button, so the code may not be generated quite conveniently. In this case, it is better to choose the last variant in advance and later change the generated code to your requirements.

1.4.5. IntelliSense reference autocompletion

With the simplest examples of tasks AI can easily complete given tasks but we overview the real world issues during development so now, our developer should supplement the existing code.

- 10) Thereafter the developer starts to add some changes in code manually using IntelliSense (Fig. 1.20).

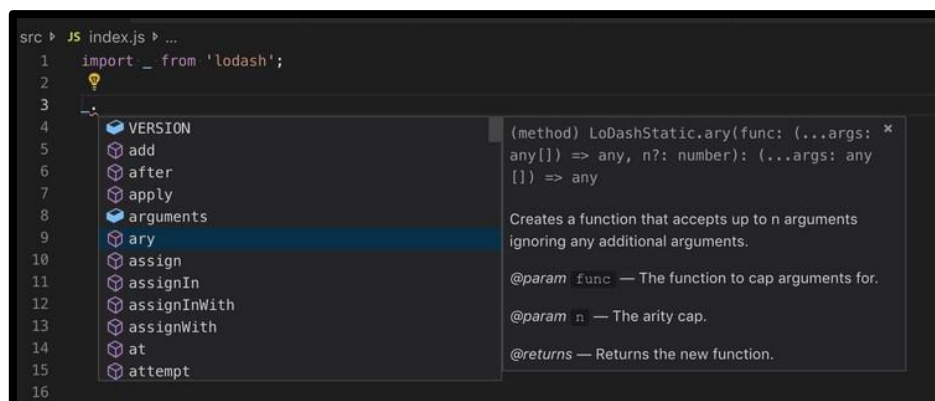


Fig. 1.20. “Code auto completion with IntelliSense”

1.4.6. Code formatter manual preferences and work

- 11) Now we complete the given task, therefore we need to format all files to developer friendly code view. The preferred Code Formatter in our case is Prettier.
- 12) Created a “.prettierrc” file in the main project with adding some preferences (Fig. 1.21.)

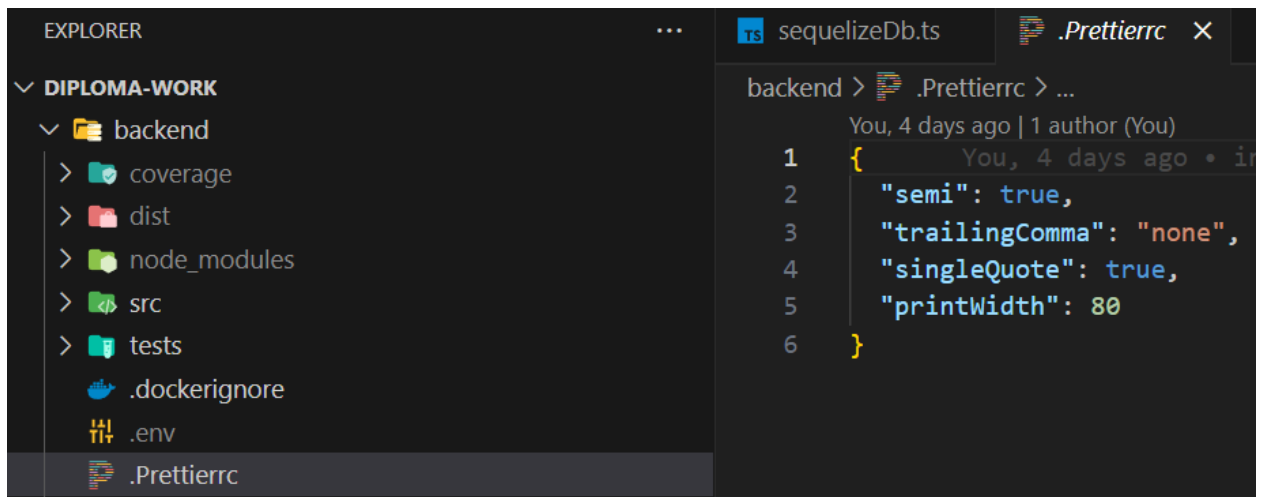


Fig. 1.21. “Importing Prettier code formatter”

- 13) Now we simply browse all unformatted files we created and push CTRL + S (Save key command)

1.4.7. Bug fixing

In the real project, generated code each time needs to be checked for bugs and if the exists we need to make next steps:

- 14) Open Copilot management system, push the fix bug on file we need to fix and AI starts to change the code (Fig. 1.22.).

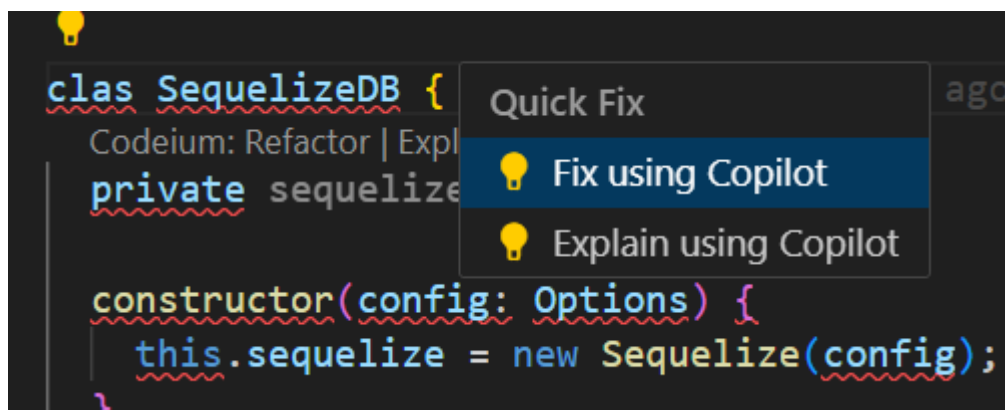


Fig. 1.22. “Bug fixing with Copilot”

1.5. Signs of methodology

Now we understand the basics of AI characteristics, how it works, what exactly it does and how to integrate all our technologies in our project developing via IDE. Also we know that there is not much difference between IDE preferences to work with these technologies.

We delved into practical studies to demonstrate the effective use of our technologies. We explored various examples of how our technologies, including artificial intelligence, can be employed in different stages of development. Here is a summary of the key points covered:

ChatGPT Request Generation - We began by defining the project requirements, identifying the need to change the simple add operation code from SQL db to NoSQL. We input this request into ChatGPT, generating the desired output that met our requirements.

Github Copilot Code Generation - Next, we utilized the output from ChatGPT and integrated it with Github Copilot. By extracting the relevant comments and instructions, we proceeded to convert them into code. We observed how Copilot provided suggestions and examples based on the comments, enabling us to generate code swiftly and accurately.

IntelliSense Reference Auto completion - To address real-world coding challenges, our developer manually made further changes to the code using IntelliSense. This allowed for fine-tuning and customization as per specific project needs.

Code Formatter Manual Preferences and Work - After completing the assigned tasks, we focused on ensuring a developer-friendly code view. We incorporated a code formatter, Prettier, by creating a ".prettierrc" file in the main project and specifying the desired preferences. By saving the files, the code was automatically formatted to adhere to the defined standards.

Bug Fixing - In real projects, generated code often requires debugging. To address this, we utilized the Copilot management system, selecting the file that

required fixing. The AI promptly analyzed the code and provided suggested changes to rectify any identified bugs.

Throughout this chapter, we witnessed the seamless integration of our technologies and their ability to assist in various aspects of development. From generating requests with ChatGPT to code generation with Github Copilot, along with manual adjustments using IntelliSense and code formatting preferences, our technologies proved to be valuable assets in the development process. Additionally, the bug fixing capabilities of Copilot showcased its effectiveness in identifying and resolving issues within the generated code.

The practical examples presented in this chapter highlight the progressive and helpful nature of our technologies, further validating their potential in model development and beyond.

1.6. Analysis of literary sources and existing analogues of AI code generators and AI code analyzers

1.6.1. Analysis of analog programs

Since the existence of a large number of software analogues was established during the study of literary sources, it will be appropriate to choose a certain number of software solution options, selected according to criteria that will most show the various qualities of the programs, their advantages and disadvantages. And therefore, first of all, it is necessary to describe artificial intelligences that perform similar actions, have any advantages or disadvantages.

It is assumed that the software improvement methodology requires artificial intelligence that can correctly respond to the necessary requests for code generation, be able to work with various aspects of software development, that is, work with structures in a given programming language, use the latest versions of the programming language, etc. .

Thanks to the software solution, the work of the developer will be simplified and facilitated, namely: creating the necessary information and minimal

documentation for creating software modules, improving the already existing code by checking it with artificial intelligence, and finding security problems in the code.

Let's move on to the consideration of artificial intelligence analogues with the aim of identifying shortcomings and advantages and using them in the development of a software solution.

1.6.2. Found software analogs to artificial intelligence Github Copilot.

Product name;

«Codeium», Exafunction, (reference in appendix A).

Basic functionality;

- 1) Generation code with AI in one software module file.

Advantages of product:

- 1) Can generate simple functions;
- 2) Can generate more complex code but with help of developer;
- 3) Complete Test generation;
- 4) Can complete refactoring but with help of developer;
- 5) Can generate comments in your code.

Disadvantages of product:

- 1) Cannot generate more complex code without help of developer;
- 2) Cannot understand the context of the project;
- 3) Cannot work with ChatGPT in your IDE;
- 4) Cannot do routine work outside the module;
- 5) Cannot complete refactoring without help of developer;
- 6) Cannot work with specific scenarios.

Product name;

«Tabnine», Dror Weiss, (reference in appendix A).

Basic functionality;

- 1) Generation code with AI in one software module file.

Advantages of product:

- 1) Can generate simple functions
- 2) Can generate more complex code but with help of developer
- 3) Complete Test generation

Disadvantages of product:

- 1) Cannot generate more complex code without help of developer
- 2) Cannot understand the context of the project
- 3) Cannot work with ChatGPT in your IDE
- 4) Cannot do routine work outside the module
- 5) Cannot complete refactoring
- 6) Cannot generate comments in your code
- 7) Cannot work with specific scenarios

Product name:

«FaulPilot», Venthe, (reference in appendix A).

Basic functionality:

- 1) Generation code with AI in one software module file.

Advantages of product:

- 1) Can generate simple functions
- 2) Can generate more complex code but with help of developer
- 3) Complete Test generation
- 4) Can complete refactoring but with help of developer
- 5) Can generate comments in your code
- 6) Can work with specific scenarios with help of developer

Disadvantages of product:

- 1) Cannot generate more complex code without help of developer
- 2) Cannot understand the context of the project
- 3) Cannot work with ChatGPT in your IDE
- 4) Cannot do routine work outside the module
- 5) Cannot complete refactoring without help of developer

6) Cannot work with specific scenarios without help of developer

1.6.3. Comparison of existing analogue programs

A comparison of the described software analogues is given in table 1.

Table 1

Comparison of code generation AI's

Property of the software	Codeium	Tabnine	FauxPilot	Gihub Copilot
Generation of simple functions	✓	✓	✓	✓
Generation of more complex code	✗ ✓	✗ ✓	✗ ✓	✓
Test generation	✓	✓	✓	✓
Understanding the context of the project	✗	✗	✗	✓
ChatGPT in your IDE	✗	✗	✗	✓
Routine outside the code	✗	✗	✗	✓
Refactoring	✓ ✗	✗	✓ ✗	✓
Generation of comments	✓	✗	✓	✓ ✗
Specific scenarios	✗	✗	✗ ✓	✓

Having analyzed all the disadvantages and advantages of analog programs, it became clear which properties of the software tool should be paid special attention to when developing a software module and what are the two main points when choosing artificial intelligence. Previously, Github Copilot only worked on a paid subscription, compared to other free AIs. Second, only Copilot can step through

the modules of the entire project to analyze code and generate code relative to existing code.

1.7. Source understanding

Based on the previous data that was researched, namely: the analysis of the capabilities of artificial intelligence, its main aspects and AI integration into the development process, a certain understanding of analogue programs and what they can do, how they differ, its advantages and disadvantages.

Thanks to the research of literary sources and existing analogs, knowledge was obtained about the problems and advantages of some analog products, which makes it possible to understand that the development of software and its use at this level requires a detailed consideration of where and how this system should work.

1.8. Relevance of the improving programs by ai code generation

1.8.1. Benefits and Challenges of AI-Driven code generation

Benefits The integration of AI techniques in development offers numerous benefits. First, it improves the efficiency of code comprehension by automating the analysis of complex code structures and reducing the manual effort required. Second, automated refactoring through AI-driven tools enhances the readability, maintainability, and modularity of legacy codebases.

Third, AI-assisted documentation generation facilitates the creation of comprehensive and up-to-date documentation that aids in understanding system behaviors and dependencies. Lastly, the use of AI models for intelligent code generation expedites the identification of critical code segments and patterns, accelerating the development process.

1.8.2. Future Directions

We propose future directions for research and development in AI-driven development. These include improving AI models' contextual understanding, addressing security and privacy concerns, developing techniques for integrating human expertise, and advancing the field of AI-assisted software analysis.

1.8.3. Formulation of the problem of using these technologies and possible solutions

1.9. Challenges

While AI-driven module creation brings significant advantages, it also presents challenges.

We address the challenges associated with AI-driven development, such as the need for accurate training data, potential biases in AI models, and ensuring the reliability and trustworthiness of generated code.

An Additional challenge is the requirement for accurate and representative training data to ensure the AI models' effectiveness and reliability. Obtaining suitable training data for development tasks may involve collecting and preprocessing diverse codebases. Additionally, biases in the training data or AI models themselves can impact the accuracy and objectivity of code comprehension, refactoring, and documentation generation. Ensuring the trustworthiness and reliability of AI-generated code and documentation is crucial to prevent potential risks or errors in the module creation process.

Conclusion

From these sections we understand that possible problems mostly appear from the point that people think that AI can simply create anything without controlling and testing the result and subsequent correction. If the developer will carefully write what he wants and fix all problems immediately, the AI really can provide several helpful tools and solutions to upgrade the module creation process to the next level.

The integration of IntelliSense, Code Formatter, ChatGPT, and GitHub Copilot presents significant potential in automating and improving development processes. The combination of these AI techniques offers benefits in code comprehension, refactoring, and documentation generation. However, challenges related to data quality, biases, and trustworthiness need to be addressed. The future of AI-driven creation of program modules lies in advancing AI techniques and fostering collaboration between AI and human expertise.

CHAPTER 2

REQUIREMENTS FOR THE IMPROVEMENT OF THE CODE GENERATION SYSTEM

2.1. Requirements for the designing of software

This work considers the generation of code using the example of an existing system for reducing links, in which a functional part of the application has already been developed by the time of working with artificial intelligence, for working with it through web requests, that is, there is no interface at the initial stage, but several tools will be developed in the future during which they will be created as well as part of product testing and changing the database used by this program. From this it becomes clear that it will be advisable to create a scheme of requirements specifically for the product being developed.

Figure 3 shows a diagram of the requirements that will be described in the current section.

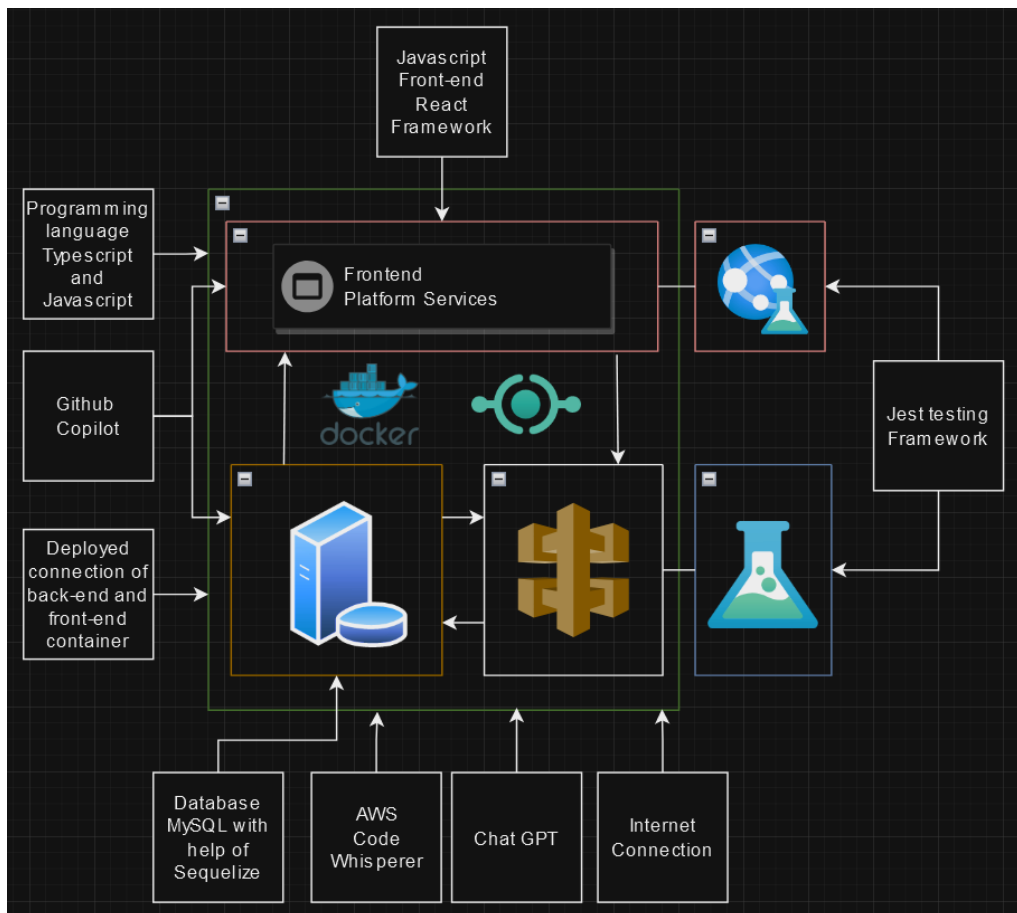


Fig. 2.1. "A model of web application design requirements"

2.2. Description of the diagram

After viewing the diagram, it is necessary to make a preliminary description of it in order to understand all aspects of the system being developed.

The field drawn in **green** is the web application as a whole, its main parts are:

- Front end part of the application
- Back end part of the application

These two parts work separately from each other and are connected to different ports, thanks to which they interact with each other by transmitting the necessary data to each other. In this model, the testing modules are separated due to the fact that they will not be included in the entire application at the production stage, so at the time of development they are necessary for the customer to receive certain data about the requirements given to the developers during the development of the software tool.

The green field is subject to improvement as a whole through code generation and vulnerability testing using artificial intelligence. In addition, there are several requirements for the green field, that is, the application being developed: writing code in some programming languages, the relationship between application components, connecting artificial intelligence both externally and during development, constant Internet access.

A component with an **orange** color, that is, a software module with a connected database, subject to partial change, due to the fact that it is necessary to change the databases with which the system works. This software module interacts with the API that formats links in a certain way, and also interacts with the front-end module of the application, that is, with the interface that the user sees. Necessary requirements for the orange field should be - a connection between the Sequelize ORM and the Postgres database.

In addition, this field should also use artificial intelligence to generate code.

The component with **white** color is the connection to the API, it receives links from the software application, sends them for conversion and sends them

back to the application in a shortened form. There are no separate requirements for it, since at the time of the start of improvement of the software solution, this part was already written.

The program module with red color is a framework interface for interaction with the user on the Internet, which must be developed. The requirements for this module are that it is written in a specific programming language, connected to artificial intelligence at the time of development for code generation. This software module is subject to complete creation as it does not yet exist at the time of software development.

Having described the diagram of requirements for the software application in detail, we will proceed to the description of functional and non-functional requirements.

2.3. Functional requirements

The field of software engineering includes writing requirements for software development. Requirements are divided into two subtypes, functional and non-functional. A functional requirement specifies a system or component of that system. It describes the functions that the program will perform. A function is described as an algorithm, input, actions, output.

2.3.1. Capabilities for backend part of web app applications

System requirements:

1. Design principles: Backend part of app needs to be written in a programming language named Typescript to use OOP design principle.
2. Testing: Backend part of app should have include library named Jest for testing the existing program modules.
3. Code: Backend program modules needs to be written with the help of AI code generation software link Copilot, CodeWhisperer.
4. Connection: Backend part of app should have connection to Front end app with fetch functions.
5. API Endpoints: The backend app should have API endpoints for retrieving and updating user data.

6. Link sending: The backend app should have functionality for taking original links from the frontend app and convert it to the shorter version and take the data back.
7. Error Handling: The backend app should handle errors gracefully and provide informative error messages to users.
8. Data Storage: The backend app should store user data securely and efficiently.
9. The program must have access to the database and the ability to communicate to provide information.
10. In a program module must be implemented to save the data entered from the front-end in the database on the backend side.
11. The program must implement the function of removing from the database the collected information from the application.
12. The program must implement the ability to change information in the database.
13. The program must have the ability to review information from the database.
14. On the main page of the web application, a medium-sized logo will be displayed, and the name of the system will be given as a large size.
15. The program should have Redis db as app cache for fast request and responses in app.

User data requirements:

1. User Authentication: The backend app should allow users to register and login to their accounts.
2. User Authorization: The backend app should have different type of authentication for preferred user choice.
3. Security: The backend app should be secure and protect user data from unauthorized access.

2.3.2. Capabilities for frontend part of web app applications

User pages:

1. System on the authentication page;

2. Admin page in Google service;
3. Main page auth requirement;
4. Input panel on the main page;
5. Data sending button;
6. Delete user from dataset;
7. Change user data from dataset;
8. View user data from dataset.

System main:

1. Development programming language requirement;
2. Each page company logo;
3. Front end app should have be written using React framework;
4. Front end app should have use code generation AI software in time of writing the program modules;
5. Sending data to back end app.

User side:

1. User Interface: The frontend app should have a user-friendly interface that allows users to easily navigate and interact with the app;
2. User Authentication: The frontend app should allow users to register and login to their accounts;
3. User Authorization: The frontend app should have different levels of access for different types of users;
4. Data Display: The frontend app should display user data in a clear and organized manner;
5. Data Input: The frontend app should allow users to input and update their data;
6. Error Handling: The frontend app should handle errors gracefully and provide informative error messages to users.

2.4. Non-functional project requirements

Non-functional requirements represent aspects of the quality of a software system; they look for standard parameters that are necessary to track the specific

performance of the system. For example, the loading speed of the site interface. Non-functional requirements necessary to ensure accurate operation without system crashes and efficiency of the entire software system. Improper support of non-functional methods can lead to unsatisfactory system operation with the user.

2.4.1. Requirements for backend part of web app applications

1. Performance: The backend part of the app should be fast and responsive, even under heavy load, caused by many requests from different users.
2. Scalability: The backend app should be able to handle increasing numbers of users and data without performance degradation or with a low degradation.
3. Reliability: The backend app should be reliable and available at all times.
4. Maintainability: The backend app should be easy to maintain and update.
5. Security: The backend app should be secure and protect user data from unauthorized access.
6. Web: The backend app should have stable connection to the internet because of api that convert the links

2.4.2. Requirements for frontend part of web app applications

1. Performance: The frontend app should have the same performance requirements as the backend part.
2. Scalability: The frontend app should have good program architecture for scalability like module addition (pages addition), design changing.
3. Reliability: The frontend app should be online all day.
4. Maintainability: The frontend app should be easy to maintain, update with new versions, etc.
5. Accessibility: The frontend app should be accessible to users with different preferred variants of authentication.
6. Usability: The frontend app should be well designed with a soft interface.
7. User Interface: The frontend app should have a well designed interactive and dynamic UI for a good User experience.

8. Design: The frontend app should have a comfortable design to interest the user to stay on the web page longer.
9. Cross-platform: the frontend app should have cross-platform between PC-Mobile interface.

2.5. Detailed Description

2.5.1. Capabilities for backend part of web app applications

System requirements:

1. Design principles: among different options for using a large number of programming languages, the Typescript language was chosen because it adheres to the principles of Object-oriented programming and also because the application being developed will not be very complex. Testing: Backend part of app should have include library named Jest for testing the existing program modules.
2. Testing: Software testing is necessary for the customer's firm confidence in the correctness of the future product without system errors. Testing also shows the code coverage that has been verified and provides a partial understanding of possible problems the system may have in the future for product support by other developers.
3. Code: During the development of the back-end part of the web application, it is advisable to use artificial intelligence, which will reduce the time for the development of software modules.
4. Connection: The connection between the front-end part and the back-end part enables the exchange between the software parts of important information needed by the users of the web application.
5. API Endpoints: The connection and exchange of data between the server part and the third-party API is vital in this case since there is no implementation of the link shortening function and instead the functions provided by the third-party product are used.

6. Link sending: The backend app should have functionality for taking original links from the frontend app and convert it to the shorter version and take the data back.
7. Error Handling: Providing errors to users when performing any actions makes it possible to understand what exactly was done wrong, or to gain an understanding of from which side the error was received, that is (whether an error was received from the server side or a regular error due to incorrect user actions, such as going to an address that doesn't exist).
8. Data Storage: Saving user data is one of the most important points when developing software; if this point was not achieved during development, this will definitely lead to people not wanting to interact with a product that loses their data, such as (data about already shortened links that the user introduced earlier).
9. Access to the database is necessary in order to capture and extract the entire array of data that will be stored in the database. This will allow the system to capture and capture information through customer requests.
10. The implemented software must be strictly configured for the rapid interconnection of the interface with which the user interacts, sending a request to the server side and dynamically processing the response to update the website page.
11. In this application, it is necessary to implement the deletion of a certain amount of data on request, as well as a cache application in the form of a high-speed database that will store the data along with their deletion timer, that is, after a certain time has passed, the data will become outdated, or, after too many requests, deletion oldest requests.
12. Changing the data is necessary if an error was made when adding information, or incorrect information was entered, or the data is simply out of date. In this case, if this information relates to user data, it is necessary to be able to change the data.

13. The ability to view information should be implemented in the software part of the interface with which the user interacts, as well as in the admin panel in which Google account data will be posted.
14. This display is made so that the user can immediately understand which company this product belongs to and in the future can quickly find other applications, while having confidence in a specific company if he liked the previous product.
15. With a large number of users using the application simultaneously, a drop in performance is inevitable. Because applications use rather slow databases, they cannot meet user requirements on a large scale. To avoid such problems, an additional database is needed that would have a fast response time but would not burden itself with a large array of data. Thus, the most frequent user requests will be temporarily stored in this database and sent in response to user requests.

User data requirements:

1. User Authentication: Logging in with your email is necessary both for authentication of a specific user and on the part of the company, which in the future could extract useful information from the array of user data without violating his rights. For example, isolating from the total mass of data information which particular email domains users use most, which would give an understanding of what aspects are worth paying attention to in the future.
2. User Authorization: Additional authentication choices are necessary because some users may have a limited amount of time to interact with the product, in which case it is necessary to implement additional login methods that will significantly save time.
3. Security: Data protection is one of the most important aspects when developing software; loss of data or its public disclosure will be a strict violation of the user's rights, which can lead to lengthy proceedings and problems from the law. Therefore, data must be protected in all possible ways at the time of development.

2.5.2. Capabilities for frontend part of web app applications:

User pages:

1. System on the authentication page should have a button to go to log in via Google authentication, so that the client can log in with an already created Google account.
2. On the admin page in Google service, the administrator should see all the users logged in with a Google account, this is necessary in order to, in some cases, be able to check user information or change it in the future.
3. Main page of the app should appear only if the user logged in, otherwise the user should be redirected to the auth panel. In this way, it will be possible to limit the ability of an unknown user to interact with the product and allow only identified users to work with it.
4. On the main page should be an input panel where the user can put the link he wants to convert, together with the information modules on the page this will give the user a quick understanding of the interface and exactly how to send links.
5. On the main page should be a button with which the user can send the link to converter API. Together with the points described earlier (modules with basic information and a data entry field), this is an additional understanding of working with this interface.
6. Deleting a user makes it possible, for some reason, to exclude a specific person from the list of users of a given product. The reasons may be different, but the main one is incorrect operation of the product or an attempt to obtain benefits or confidential information from the application.
7. Changing user information may be one of the requirements of the user himself if he has entered incorrect information about his account. There may also be situations when previously valid information is already outdated and has been changed and because of this the user can no longer access this application.

8. Adding a user by the application administrator can also be implemented for the reason that the user cannot create an account himself but also wants to use the system.

System main:

1. You can implement a logo in the application in a dynamic way, this will make it possible to add a logo on each page, regardless of its content.
2. Since the application does not require a complex architecture, it is advisable to use the easiest programming language for web application development, which is called Javascript, in this way it will be possible to implement a website quickly and efficiently.
3. Using the React framework in combination with the Javascript programming language will make it possible to create dynamic, scalable web applications with an architecture convenient for further development, which will give more options for changing the product after the release of the initial version.
4. Development of the front-end part of the application using artificial intelligence that generates code will make it possible to speed up the software development process due to the fact that the developer will eliminate the part of developing a complete architecture from scratch and will only gradually supplement the proposed parts of the generated code.
5. The user will be required to have an original link, which is subsequently converted into a short version of it, but at the same time makes it possible to follow it in the same way as the original one.

User side:

1. User Interface: A user-friendly interface and easy to understand makes it possible to quickly navigate through parts of the application and obtain the necessary information through simple interaction.
2. User Authentication: Logging into your accounts is necessary because the user may return to the application and interact with it again after a while. Such a system must also be implemented due to the fact that nowadays every application inherits this system behavior.

3. **User Authorization:** Different access levels separate layers of interaction available to different types of users to differentiate access options and also provide a better security system.
4. **Data Display:** A structured display of data, for example from older to newer, provides a clear understanding and comfortable interaction with the system in the future after a certain amount of work has been done in the system.
5. **Data Input:** Entering and updating data allows the user to perform lengthy actions in the system without receiving errors or delays in response.
6. **Error Handling:** Providing a correct description of errors or displaying a system boot icon under load will make it possible for the user to understand that he continues to do everything correctly, but due to the large number of simultaneously executing requests, the system processes them and requires more time to respond.

2.5.3. Requirements for backend part of web app applications

1. **Performance:** Fast responsiveness must be realized through the use of a variety of technologies and systems that allow this to be done. Starting from a high-speed database in the form of a completion application cache to correctly configured and connected program modules for quick interaction of the system with the user.
2. **Scalability:** Deterioration in performance is inevitable since even the server on which the application was running may not be able to work quickly under a huge load, however, such problems must be resolved in a timely manner by moving the system to more productive servers or searching for possible problems in the system architecture. It is also possible to locate parts of the system on different servers to try to gain benefits in overall system performance.
3. **Reliability:** Supporting robots of the system rests on the server. The owner of the system must promptly agree on such support (in our case, 24/7), since without this the system will be susceptible to possible errors in the event of an

emergency restart of the servers and even loss of the latest data that was entered when the application was stopped.

4. **Maintainability:** Correct implementation of the architecture and mobility of the application will provide its bonuses with further changes to the system in the form of easy configuration or adding software modules.
5. **Security:** Unauthorized access can be obtained as a result of system hacking. To avoid such cases, the architecture of software modules responsible for security should be built in the most secure manner.
6. **Web:** Internet access is one of the most important aspects when sending an application to production, which will make it possible to interact with the system as a whole. Without it, such a system will not be able to update or save user data, not to mention the main function - converting links.

2.5.4. Requirements for frontend part of web app applications

10. **Performance:** An unambiguous requirement during development must be the correct implementation of systems that will enable the application to interact between its software parts correctly without errors.
11. **Scalability:** An architecture that combines the best principles and rules for building a system will make it possible to support, change and delete software parts of the system without any problems.
12. **Reliability:** For user convenience, system support is required that will work around the clock.
13. **Maintainability:** To update versions when developing a system, you must configure and use version control, which will make it possible to quickly switch between versions with new or old software modules and then send a new version of the system to the server.
14. **Accessibility:** Access to different authentication options is necessary since the user may refuse to work with the system for any reason.
15. **Usability:** Since the initial version of the back-end application assumed the use of the system as a simple API, it is necessary to take into account the

software modules that must be changed in case of transition to a full-fledged full-time application.

Interface:

Since this application is aimed at many users, the interface should be clear and pleasing to the eye. To make the system as comfortable as possible, you need to reduce: the amount of information in the form of buttons and information columns, as well as the excessive number of colored parts of the page.

Regarding pages with basic information, such as a main page with a link field, it is necessary to reduce the amount of miscellaneous information other than the general operation of a list of shortened links by the user.

From here it is clear that in addition to the panel where the user enters information, below there should be a dynamic display of links that the user has already used in convertible form. Thus, a simple web application interface needs to be developed, which should be created as follows:

- 1) The correct part with the navigation bar.
- 2) On the navigation panel, on the left is a logo centered on a link with buttons, and on the right is a user icon with his mail name.
- 3) In the middle, a panel with general information that the user can perform on this page.
- 4) Below is the field for entering the necessary information.
- 5) Even lower is the button for sending information to the server side.
- 6) Under the main container there should be a dynamic generated list that displays all the converted links that the user sent, and so that the next links sent by him do not erase the previous ones.
- 7) At the very bottom there is a container with contact information, the company's rights to this product, and some links.

Design:

The design of the interface of the web application is a very important part of the software, because of which users may have a great need for this software solution. In the event that the interface is unfriendly and difficult to master with a

50 percent chance, the user will immediately change his mind about using the product.

To achieve such design criteria, it is necessary to use a number of technologies during the development of the web part of the application.

In addition to the standard need for a markup language and styling with the help of HTML and CSS, it is also possible to use the JavaScript programming language and a framework for it called React, this will add responsiveness and dynamism to the application, i.e. web pages will respond to the user's actions and immediately change according to his needs.

Also, for greater necessity, it will be possible to use such frameworks as Material UI which give already developed front-end solutions etc.

Cross-platform:

Given that the product being developed must be open to all users, the need for a cross platform immediately becomes clear. That is, the application works both on PCs, tablets and phones. This will add a large number of users, as web applications are increasingly being used through phones.

Conclusion

The section reviewed the basic requirements for the design of the web application that will be developed. A study of aspects of the program was made, the main parts of the program were separated into program modules, to obtain the final form of the finished product.

Pitfalls that must be taken into account when describing the requirements for this software are described.

Revised user interface module, database connection module and API that allows you to receive formatted links, API module that receives original links and converts them into a shortened form.

Received information about external requirements that are necessary during development A description of the functional and non-functional requirements for the concrete system was made based on the preliminary research.

Each requirement is highlighted with its detailed description for comprehensive information.

CHAPTER 3

PROPOSED ARCHITECTURE OF THE SOFTWARE APPLICATION

3.1. Software architecture

When building a "shortening of web links" system using several products that perform artificial intelligence actions, it is advisable to use the MVC pattern. This software construction methodology includes a model (a description of abstract objects that will be used later as tables in the database), a representation (a system layer designed to display the user interface and general interaction with the system), a controller (controllers for performing certain actions in the system, data redirection, etc.). Since the web application chosen for software development will be a web application, the use of this pattern will help the developer to speed up the solution of fundamental problems during application development.

A necessary parameter for the implementation of the system will be the addition of several databases to it.

First, the system should have a standard slow database for handling all data such as user models and links.

Secondly, there should also be a fast database that will be responsible for saving a small number of records that can be quickly corrected in response in order to improve the performance of the web application.

In the first case, the Sequelize cross-platform database connection system will be used. Its advantage is that it provides an opportunity to quickly change the type of Database without problems, as well as an easy way to connect these databases. Since when using our AI code generation methodology, the documentation created chose to change the database from MySQL to PostgreSQL, the final database will be the latter.

In the second, a high-speed Redis database will be connected to the main operation of the system in the form of a wrapper over the backend part of the application. This will allow you to pre-search for a record in Redis and issue a

response, or if the necessary data is not found, the next work will already be done with a regular database.

Thanks to the ORM system implemented today, the software supports such a scheme by creating a Sequelize model (standardized Sequelize object) in the program, which has the usual data fields, methods and a constructor. Such a Sequelize entity should not have any additional complicated functions, since complex operations with the database will rely on Service (service) classes and methods, and all system logic, redirection, use of necessary functions and APIs will be performed by Controller objects, in which it will be possible to create additional solutions.

The diagram of the MVC pattern is shown in the figure 3.1.

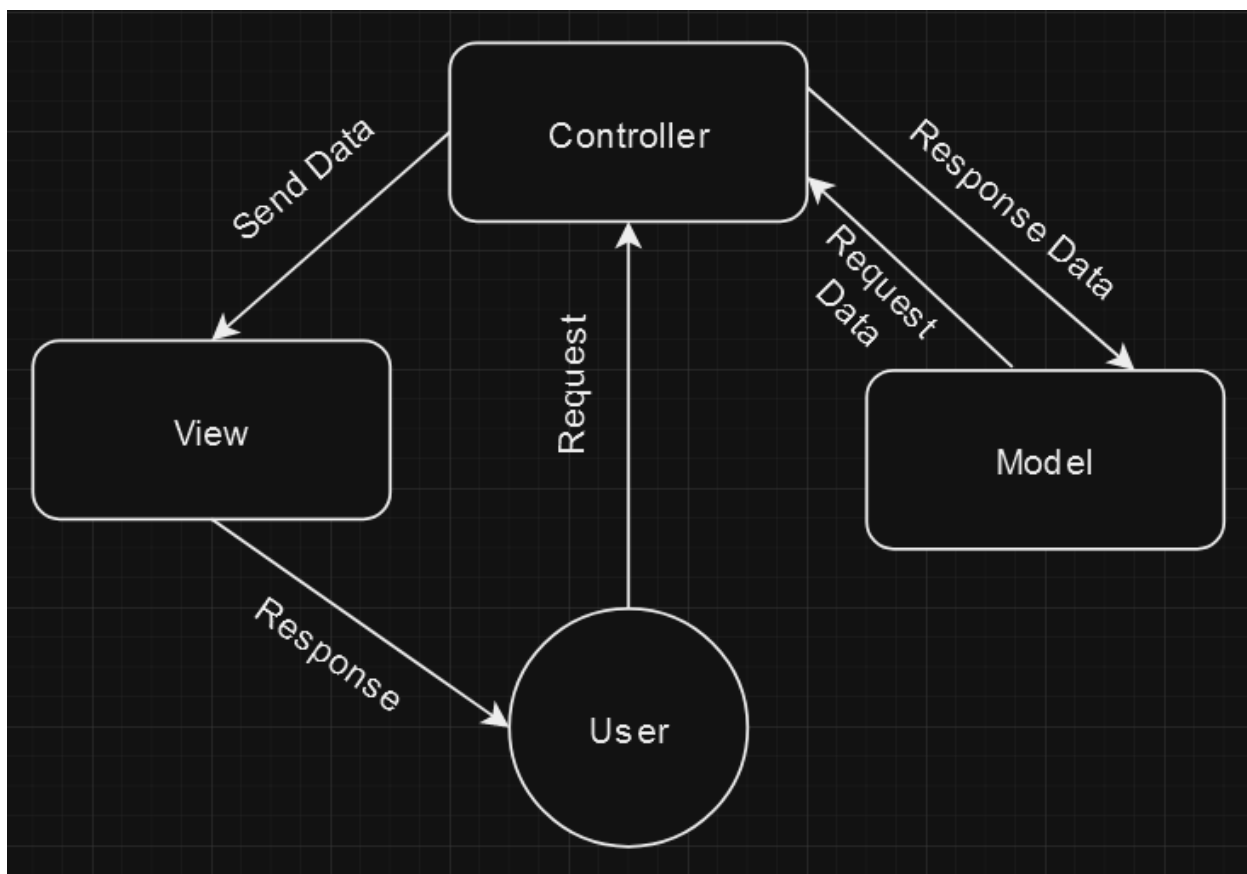


Fig. 3.1. “Scheme of the MVC pattern”

Using the selected software architecture pattern, it is appropriate to also use the concept of data processing CRUD (abbreviation from the English abbreviation of four words: create, view, restore, delete). This concept determines that the

system will calculate data processing according to certain operations. Creation, editing, modification and deletion of data.

Figure 3.2 visually presents the functions of the concept.

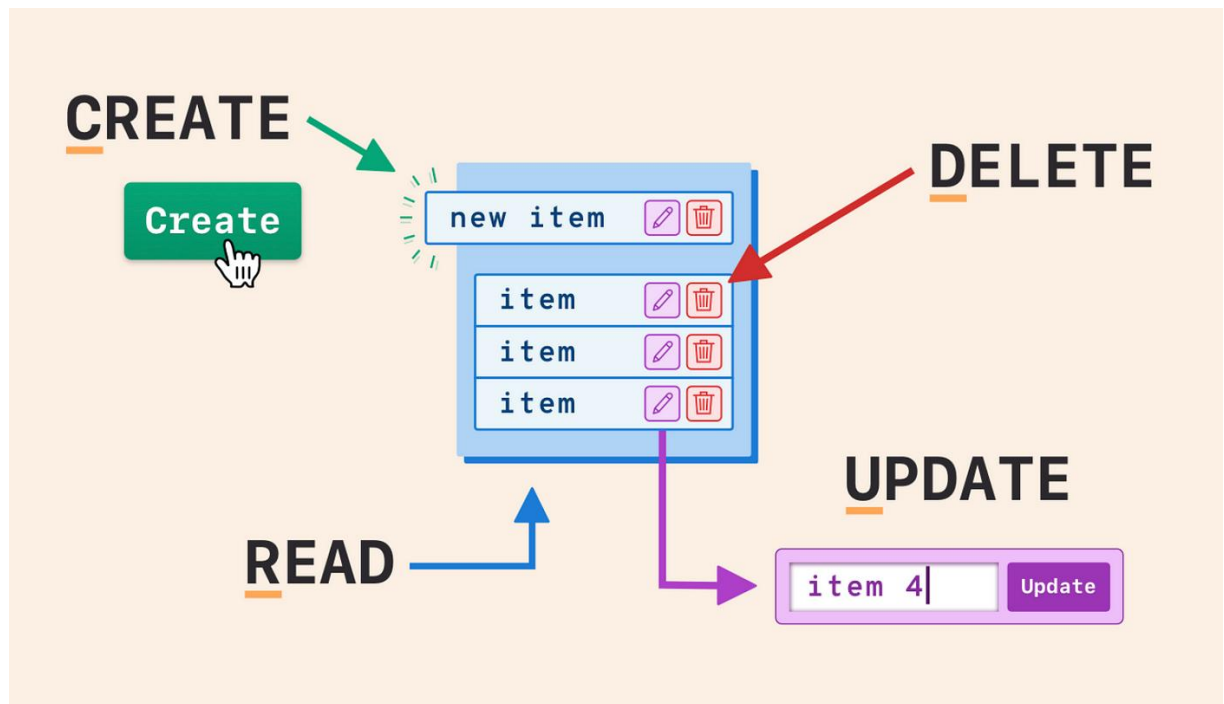


Fig. 3.2. “Functions of the principle of C.R.U.D.”

3.2. Diagram of the deployment of the link shortening web application

Installing the system will not have any complicated steps. The created deployment diagram of the system will help to understand aspects of its operation. It is assumed that the finished product will be installed on the server of the customer's company, after which it will work as a separate product, that is, a separate web service. The use of this system may vary depending on the needs of the end users.

One possible option is to use the system to create a large number of links that can be followed, embedded in documents, web data tables, etc.

Regarding ensuring the reliability of the system and the processing capacity of the web application, it is necessary to introduce a strong emphasis on security. During development, the protection of user data and the links they send must be monitored at the highest level.

The graphic appearance of the system will meet all needs in accordance with the UI and UX principles of system construction, which will allow smog to use the web application quickly and comfortably.

For a more detailed description of the system, a deployment diagram should be presented

Figure 6 shows the deployment diagram..

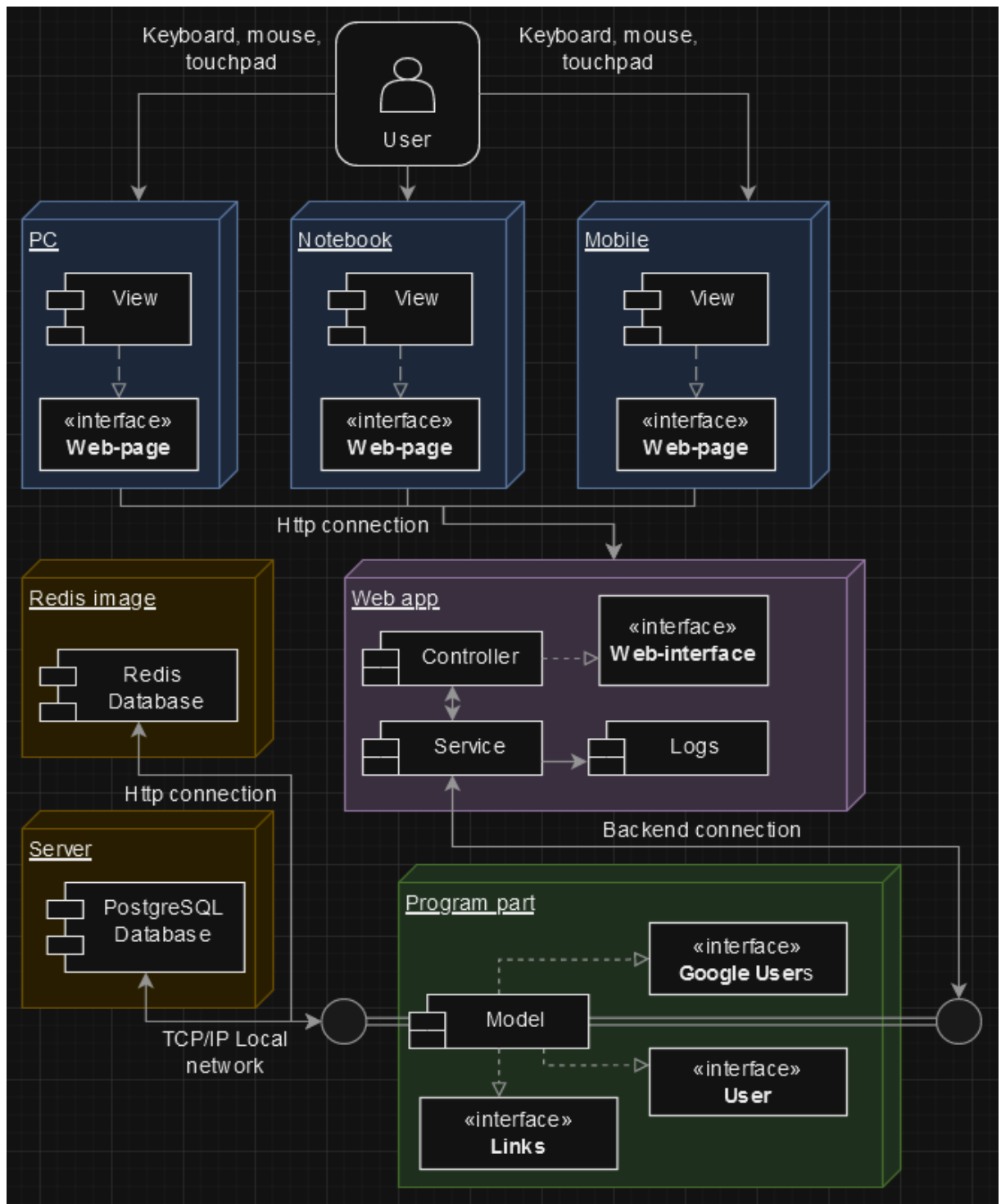


Fig. 3.3. "System deployment diagram"

After looking at the diagrams, it is possible to characterize several important aspects of the system.

This system contains three components of a blue color. It should be noted that users can use three types of hardware devices. This can be a corporate or home computer, laptop or smartphone.

In the next ball, which is connected between the blue ball and the purple ball, whatever browser is installed on your hardware, you can interact with the system and see the type of network connection at the interface modules when performing operations with the system.

The purple color web add-on interacts with two databases through a front-end controller that responds to the user's actions and after the service module, which also interacts with the databases themselves.

Application of software solutions under the hour of development of the service module use description of the abstract model of the user, send, etc. The connection with the databases is completed under a logical mind. In case of a different version of the first database, the data continues to interact with the static other database.

For a more detailed explanation, we need to take a look at the component diagram

3.3. Component diagram of the link shortening web application

The details of the relationships between the components in the system can be viewed when the component diagram is expanded. The display component in the class module allows you to identify different warehouses throughout the system, which determines what they are responsible for. Figure 3.4. is a diagram of the system components is presented with the ability to demonstrate how operations are performed on the skin of the program.

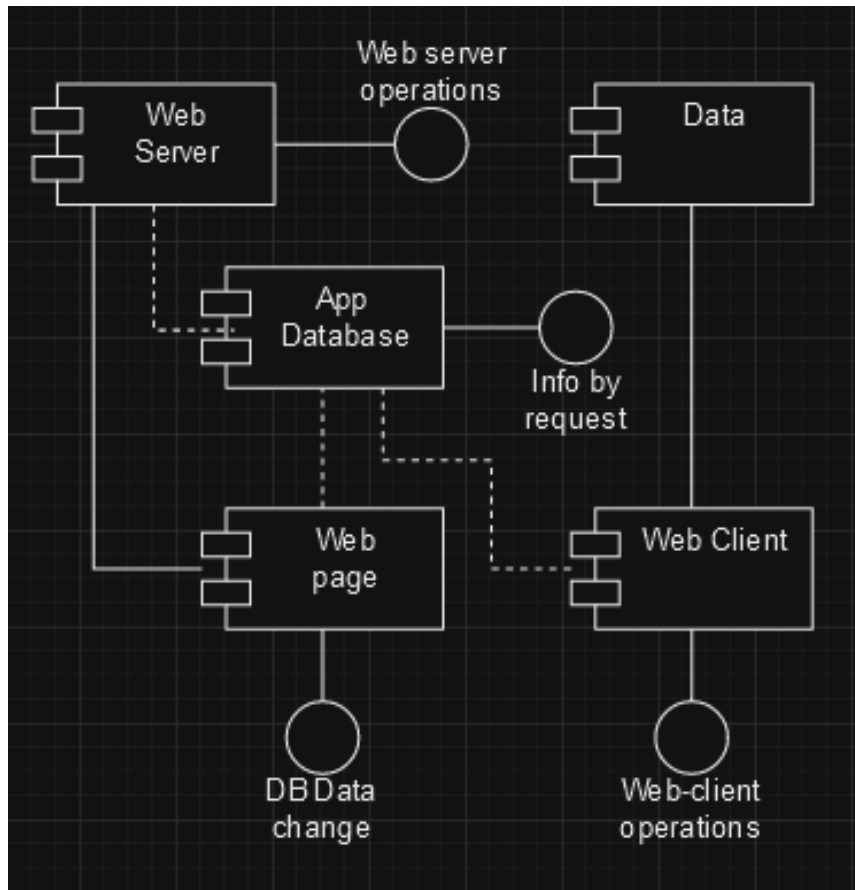


Fig. 3.4. “Diagram of system components”

3.4. Class diagram

The class diagram was created for the back-end part of the web application. We will review the diagrams for the front end part later. On the basis of abstractions and their connections, it is possible to see the relationship between system components.

A large volume of fields has been written to map Sequelize to an object (that is, a certain model created on the basis of the Sequelize class), thanks to which the system initializes these models and creates the corresponding tables in the database

It makes no sense to describe the functions and class constructors in detail because they perform standard actions that are subject to this type of functions. Their basic logic can be immediately understood from their name.

However, it is necessary to describe some unique functions that do not follow standard methods and solutions:

- Router - this class performs actions on receiving a request from the front-end part, and the corresponding redirection of the request to the corresponding methods of the controller when a response is returned in the form of data or a status of 200 OK, the front-end part considers this request to be fulfilled correctly, stores and/or displays the corresponding data.
- Controller - performs both normal functions corresponding to it for redirection to the Service part and unique ones, such as sending links to a function that will communicate with a third-party API and perform the necessary actions.

Service classes have a characteristic logic, namely:

- Creation of entries in the database
- Deleting entries in the database
- Changing entries in the database
- Receiving data from the database

These actions are performed according to a normal user, a user from the Google service, Links (two types in one row of the database table: abbreviated and original). Generate a hash code to generate the corresponding encryption hash in the Redis database for further quick lookup in the table.

Classes are not divided into interfaces because they do not have possible changes for future needs at the time of the technical task.

The User class is intended for use when working with the system as a unique user, necessary during authentications in order to understand that specific actions are performed by that user. According to him, the system modules will display unique responses based on his actions. After passing through the security safeguards of the system, a coded token is created for each user, which later enables the user to enter the main page of the system and perform some actions. Without a token, the user will not be able to access the pages, as the token is checked on each page.

The GoogleUser class (google service user) performs a similar function to an ordinary user, but with some differences, because the google service system

gives partial information about the user when he chooses the type of authentication on our platform using google, instead of a password, the google service also sends an authenticated token, which will be systemically verified using the appropriate functions, after which this user will be entered into the database just like a normal user and will accordingly receive a token of our web application.

The Link class is the main class for the system because the entire application is built around it. Performs the role of receiving a link from the user, which will be converted using a third-party API and in turn entered into the database as a single row with two columns, the original link and its shortened version.

The class whose names contain the Service postfix is responsible for connecting and working with the database. In addition, the system has three main classes of configurations that help the software work at the lower levels of the program code and without which the entire system would immediately increase several times.

Sequelize ORM handles all the work between the created abstraction models and the tables in the PostgreSQL database. In addition, the ORM receives data for connecting to the database system itself, i.e.: connection type, administrator login, password, database name, its specific schemas using a configuration file.

React configures the web part of the application. It configures exactly which pages of the presentation will be read, their format, the location of the main HTML file, and changes of interface components according to user actions. With the help of fetch functions, as a complex unit, the web part also communicates with the back-end part for some actions.

For each of the solutions, that is, the backend and the frontend, there is a configuration file, it is hidden from viewers and is a system file that is needed to maintain the most important data of the system itself, without which the ability to send requests between software solutions, communicate with a third-party API, etc. will disappear.

Using this architecture, the simplest work with data is obtained in the developed software. Figure 3.5 shows the class diagram of the system.

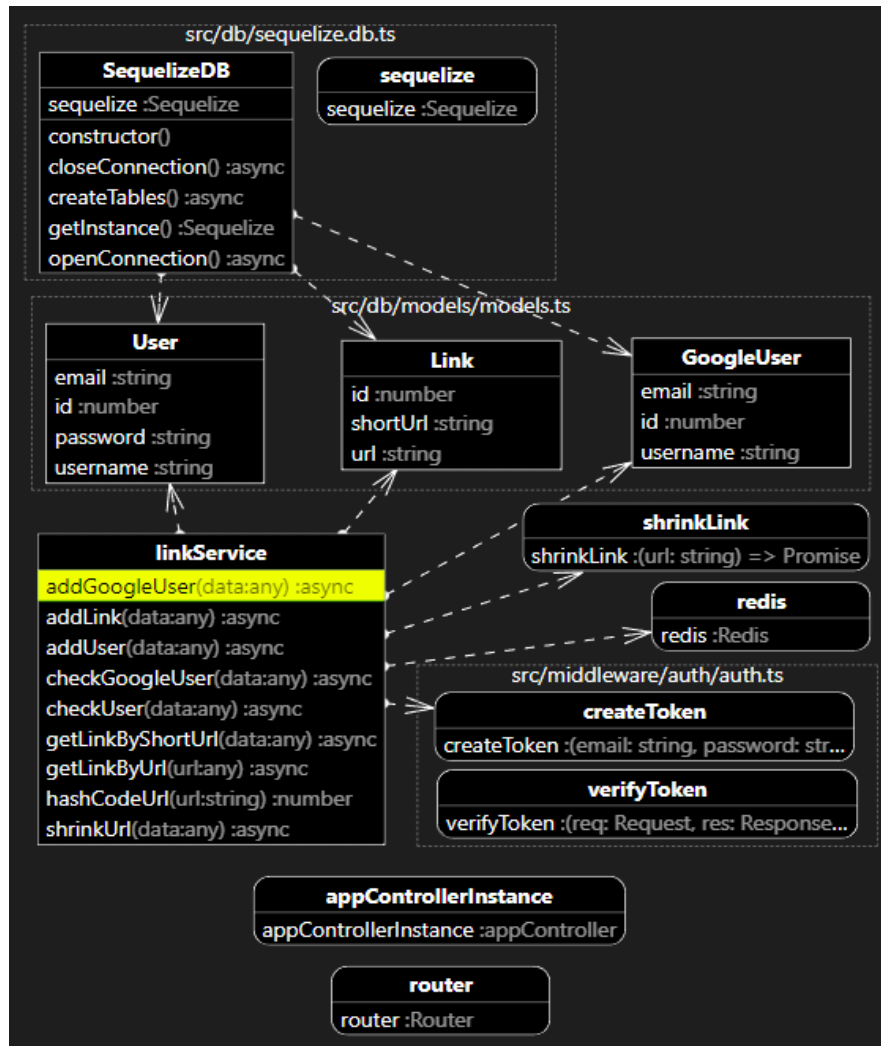


Fig. 3.5. “Class diagram”

3.5. Construction of connections of physical equipment

Let's move on to the description of the system at the physical level. Thanks to the scheme of working with the system, it becomes clear how the system should be used and how it interacts with each other. Any number of users can use the Internet and go to the resource through a link or with the help of browser search engines, and then start interacting with the system itself. An important factor of this system is that, after its implementation, it will be configured as separate systems that interact with each other, they will be uploaded to some server according to the customer's needs. Thus, there are three containers that are programmed together and can be replaced as needed:

The first container is the front-end solution, i.e. the user interface, it starts on its port, and communicates only with the second container, i.e. the back-end part, using the configured internal network.

The second container is the back-end part, it will have all the logic of interaction, and the necessary mechanisms without which the first container will not be able to process the data necessary for the user, but will only be an empty display of the interface without interaction with the system. Also, a common database will be locally configured in the back-end part to store information.

The third container is a separate high-speed database the backend part will interact with it before working with the main database. Interaction will be performed thanks to http requests, just like in a local network.

In general, such a system will allow implementing the principle of modularity, when some parts of the system can be used for other software applications in the future, which will speed up the development of new systems.

Figure 9 shows the construction of connections of physical equipment.

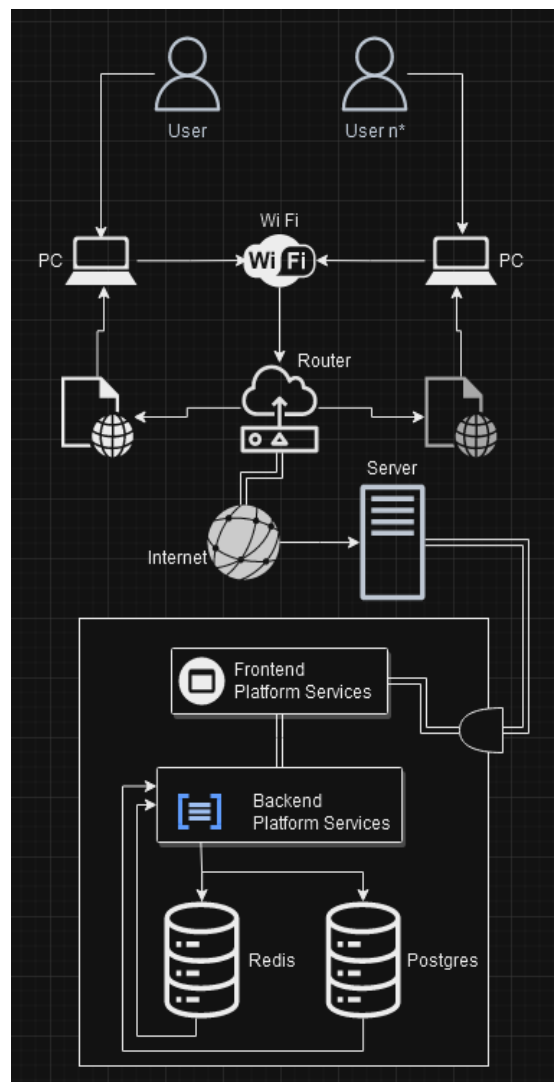


Fig. 3.6. “Construction of connections of physical equipment”

Conclusion

In this section, the diagrams and schemes built on the basis of the completed software solution were described and considered. First, the main principles and patterns by which the system was built were taken, and it was considered which aspects should be paid attention to during development.

Then a system deployment diagram was created, which gave an accurate understanding of the software modules, their use and interaction with each other.

In addition to it, a component diagram was also developed for a better understanding of the internal mechanics of the system.

For a visual understanding of the classes and methods of the system, a class diagram with a detailed answer to the step-by-step implementation of the system was presented as an open source code.

At the end, the possible physical implementation, implementation and use of the system by the user was investigated.

CHAPTER 4

PROTOTYPE IMPLEMENTATION USING THE CODE GENERATION METHODOLOGY

4.1. Development of a prototype of a web service for shortening links using the methodology of using artificial intelligence

According to the provisions, when developing software by some company under the order of an individual, it is necessary to take into account and respond to the needs of the customer. Thus, after receiving a certain set of documents, such as a technical task, use cases (use cases created on the basis of a technical task and a study of how the user will use the finished system), needs and the deadline by which the manufactured product must be delivered (divided into smaller parts of the work with phased deadlines) it is possible to produce a reliable and high-quality product.

Based on obtaining such information, the first factor is cooperation with the customer and characterization of the appearance of the system (in the usual case, it is a layout, interface of the future system). After that, this data is sent to the chief specialist in software architecture. It defines and sends all the necessary questions about the functionality of the system. These additional aspects are discussed with the customer, and after some time of communication, a final agreement is drawn up on what exactly will be developed. From this moment, the development of the prototype begins.

There are many software development methodologies and models, so it's worth noting that prototyping and using these methodologies will vary from system to system.

A prototype is a pre-made product that can have software modules that are subject to improvement and change. Depending on the contract with the customer, the type of prototype may be different. After the production of the previous product, the customer confirms or adds information about which modules will be changed. It is correct to note that the prototype can fully satisfy the needs of the

customer and immediately be transferred to the production version after verification by methods of system testing.

According to the listed information, the selected link shortening system in the form of a web service will be developed on a server to which the customer will have access, thus he will be able to view real-time scaling of the system.

4.2. Prototype of the developed system

App will be separated into three parts, each part have different ports and connections preferences. Front-end part will be running on local host and port 3000 (default port for frontend application). Back-end part using Postgres db which running on port 5432, and the app on 8080 (default port for backend application). The third part will be redis container on port 6379. All parts are different containers connected by one network with Docker system.

The beginning of the software begins with a description (characterization) of the models that will be used in the system. For which classes are implemented that are inherited from another base class “Model”. This class is built into the backbone of the ORM system by Sequelize, a package whose framework contains a wide range of classes for initializing models. Once again, it will be possible to quickly create tables and data from the selected databases.

The necessary models for the system were equipped with the following abstractions:

- Link
- User
- GoogleUser

The “Link” model is aimed at selecting a table with messages that will be sent to the correspondent and sent messages that will be removed after conversion for a short view and return to the customer.

The “User” model is oriented towards the use of a user account. Without this model, the system will lost its ability to create a user account as well as entering under a specific user account for further access to additional functions.

The “GoogleUser” model has similar logic to the previous model. The only difference is that the Google account returns to a third-party resource and may go through a full verification of these returns.

This file use DataTypes for creation of model provided by Sequelize framework

Listing 1 shows these models in action.

Listing 1

File “Model.ts”

```
import { DataTypes, Model } from 'sequelize';
import { sequelize } from '../sequelize.db';

export class GoogleUser extends Model {
  id!: number;
  username!: string;
  email!: string;
}

GoogleUser.init(
  {
    id: {
      type: DataTypes.UUID,
      defaultValue: DataTypes.UUIDV4,
      primaryKey: true
    },
    username: {
      type: DataTypes.STRING,
      allowNull: false
    },
    email: {
      type: DataTypes.STRING,
      allowNull: false
    }
  },
  {
    tableName: 'GoogleUser',
    sequelize: sequelize
  }
);

export class Link extends Model {
  id!: number;
  url!: string;
  shortUrl!: string;
}
```

```

Link.init(
  {
    id: {
      primaryKey: true
    },
    url: {
      type: DataTypes.STRING,
      allowNull: false
    },
    shortUrl: {
      type: DataTypes.STRING,
      allowNull: false
    }
  },
  {
    tableName: 'Link',
    sequelize: sequelize
  }
);

export class User extends Model {
  id!: number;
  username!: string;
  email!: string;
  password!: string;
}

User.init(
  {
    id: {
      type: DataTypes.UUID,
      defaultValue: DataTypes.UUIDV4,
      primaryKey: true
    },
    username: {
      type: DataTypes.STRING,
      allowNull: false
    },
    email: {
      type: DataTypes.STRING,
      allowNull: false
    },
    password: {
      type: DataTypes.STRING,

```

```

    allowNull: false
  }
  tableName: 'User',
  sequelize: sequelize
}
);

```

Configuration files that will be used by these models in the future to connect several databases. This configuration file is responsible for connecting the Redis database, a high-speed database that will be connected to the system by creating a container based on the downloaded Redis package snapshot in Docker. it has stable tables with data only connected by hashed ids.

File use redis feature received from “ioredis” package as well as “dotenv” feature from “dotenv” package. Dottedenv feature gives ability to take some secured credentials for our app

On listing 2 presented the connection between software and the image of redis.

Listing 2

File “Redis.db.ts”

```

import dotenv from 'dotenv';
import Redis from 'ioredis';
dotenv.config();

const redis = new Redis({
  host: process.env.REDIS_HOST || '',
  port: parseInt(process.env.REDIS_PORT || '6379')
});
redis.on('error', (error: any) => {
  console.error('Redis connection error:', error);
});

export default redis;

```

It is also a configuration file responsible for configuring the Sequelize ORM and its connection to the PostgreSQL database. Its advantage is that the database is arranged as a local connection and not a snapshot of a specific software that is launched separately, and therefore software settings are required to start the

database, to interact with it, to disable it, etc. In addition, tables in the database are created in this file according to the corresponding rules described in our application, as well as according to the corresponding model fields.

This file uses the dialect feature from sequelize package for creating a valid type of database to create right settings for sending data between concrete databases.

Also use User, GoogleUser and Link classes exported from model file in the app

Listing 3 describes the sequelize configuration file.

Listing 3

File “Sequelize.db.ts”

```
import { Dialect, Sequelize } from 'sequelize';
import { User, GoogleUser, Link } from './models/models';
import dotenv from 'dotenv';
dotenv.config();
const env = process.env.NODE_ENV || 'development';
export class SequelizedB {
  public sequelize: Sequelize;
  constructor() {
    this.sequelize = new Sequelize({
      database: process.env[`_${env.toUpperCase()}_DATABASE`] || '',
      username: process.env[`_${env.toUpperCase()}_USERNAME`] || '',
      password: process.env[`_${env.toUpperCase()}_PASSWORD`] || '',
      host: process.env[`_${env.toUpperCase()}_HOST`] || '',
      port: parseInt(process.env[`_${env.toUpperCase()}_PORT`] || '', 10),
      dialect: (process.env[`_${env.toUpperCase()}_DIALECT`] || '') as
Dialect
    });
  }
  public getInstance(): Sequelize {
    return this.sequelize;
  }
  public async openConnection() {
    try {

      await this.sequelize.authenticate();
      console.log('Connection has been established successfully.');
```

```

        console.error('Unable to connect to the database:', error);
        return false;
    }
}
public async closeConnection() {
    try {
        await this.sequelize.close();
        console.log('Connection has been closed successfully');
        return true;
    } catch (error) {
        console.error('Unable to close the database connection:', error);
        return false;
    }
}
public async createTables() {
    try {
        await User.sync();
        await GoogleUser.sync();
        await Link.sync();
        console.log('Tables have been created successfully.');
```

```

        return true;
    } catch (error) {
        console.error('Unable to create tables:', error);
        return false;
    }
}
}
const sequelizeDB = new SequelizeDB();

export const sequelize = sequelizeDB.sequelize;
```

These models are already used in the lost file responsible for sending commands and data together to the database. Initially, the data is sent to the usual database during user registration. After that, when interacting as a user, the corresponding model of "links" is sent to two databases.

This is necessary in order to reduce the load on the system in the event of a user request for data, because the last most popular links that were sent by users could be quickly found in the system and returned, if the data is not found in the database with a high-speed connection, the interaction is switched to a regular database that will search for certain data.

File using different types of features. Redis feature used for connection between service file app and Redis image. Dotenv for taking secured credentials from .env configurative file. createToken feature, used to create tokens which need to be used in a session of a user and know that some user has a valid authentication.

OAuth2Client feature gives our app ability for authentication for google users with their accounts.

Listing 4 describes the DAO service file linkService.

Listing 4

File “LinkService.ts”

```
import shrinkLink from '../api/shrink-api';
import redis from '../db/redis.db';
import dotenv from 'dotenv';
import { createToken } from '../middleware/auth/auth';
import { OAuth2Client } from 'google-auth-library';
import { User, GoogleUser, Link } from '../db/models/models';

const client = new OAuth2Client(process.env.GOOGLE_CLIENT_ID);
dotenv.config();
export class linkService {
  async shrinkUrl(data: any) {
    try {
      const shrunkedUrl = await shrinkLink(data.url);
      data.shortUrl = shrunkedUrl.result_url;
      console.log(
        'data in the append queue' + '\n' + data.url,
        'and ' + data.shortUrl
      );

      await this.addLink(data);
      return data.shortUrl;
    } catch (error) {
      console.log(error);
    }
  }
  hashCodeUrl(url: string) {
    let hash = 0;
    for (let i = 0, len = url.length; i < len; i++) {
      let chr = url.charCodeAt(i);
      hash = (hash << 5) - hash + chr;
    }
  }
}
```



```

        hash |= 0;
    }
    return hash;
}
async addLink(data: any) {...}
async getLinkByUrl(url: any) {...}
async getLinkByShortUrl(data: any) {...}
async addGoogleUser(data: any) {...}

async checkUser(data: any) {
    try {
        const user = await User.findOne({
            where: {
                email: data.email,
                password: data.password
            }
        });
        if (user) {
            const token = createToken(data.email, data.password);
            const userAndTokenData = {
                username: user.username,
                token: token
            };
            return userAndTokenData;
        } else return false;
    } catch (error) {
        console.log(error);
    }
}

async checkGoogleUser(data: any) {...}

export default new linkService();

```

This file is responsible for the manipulation and execution of processes necessary for the operation of the system. Includes various operations. From receiving and further sending data for processing by a service file and database, to sending data to a third-party API used by this application and further converting links. It also includes user verification and subsequent creation of tokens for data

access. It uses the Request and Response features taken from Express framework. All these methods working by receiving the tasks by request and then doing some logic according to the name of methods.

Listing 5 describes the controller file appController.

Listing 5

File “appController.ts”

```
import linkServices from '../services/service';
import { Request, Response } from 'express';
import { error } from 'console';
import dotenv from 'dotenv';

dotenv.config();
const _host = process.env.FRONTEND_HOST;
const _port = process.env.FRONTEND_PORT;
class appController {
  shrinkUrl = async (req: Request, res: Response) => {...};

  addLink = async (req: Request, res: Response) => {...};

  getLinkByUrl = async (req: Request, res: Response) => {...};

  getLinkByShortUrl = async (req: Request, res: Response) => {...};

  addUser = async (req: Request, res: Response) => {...};

  checkUser = async (req: Request, res: Response) => {...};

  checkGoogleUser = async (req: Request, res: Response) => {...}

  const appControllerInstance = new appController();

  export default appControllerInstance;
```

API file used as connection and configuration file for taking links and sending this data to shrink api as requested. This file uses the “Axios” package that changes the simple fetch function used for sending http requests and makes fast

responses with data. Here we need to create options variable valid “Axios” requests.

File using the encodedParams var created by URLSearchParams feature for creation of url requests in valid form.

Listing 6 describes the api management file “shrink api”.

Listing 6

File “shrink-api.ts”

```
const axios = require('axios');
const encodedParams = new URLSearchParams();
import dotenv from 'dotenv';

dotenv.config();

const options = {
  method: 'POST',
  url: process.env.API_URL,
  headers: {
    'content-type': process.env.CONTENT_TYPE,
    'X-RapidAPI-Key': process.env.X_RAPID_API_KEY,
    'X-RapidAPI-Host': process.env.X_RAPID_API_HOST
  },
  data: encodedParams
};

const shrinkLink = async (url: string) => {
  encodedParams.set('url', url);
  try {
    const response = await axios.request(options);
    console.log(response.data);
    return response.data;
  } catch (error) {
    console.error(error);
  }
};

export default shrinkLink;
```

The router file. Provide all routing features for frontend part of app. Manage all request and switching between routes according to some url string received from react app. Next it will send this data to some controller function which will make some actions and return the status of concrete operation. Router

can send some output status of some operation for frontend app to see some changes.

Before redirecting to some methods all routes require verification of cors rule as well as user token check. If the verification returned true value it can continue making required functions.

Listing 7 describes the route system of “app router”.

Listing 7

File “app-router.ts”

```
import express from 'express';
import linkController from '../controllers/app-controller';
import { verifyToken } from '../middleware/auth/auth';

export const router = express.Router();

router.post('/login/user', linkController.checkUser);
router.post('/register/user', linkController.addUser);
router.post('/auth/googleuser', linkController.checkGoogleUser);
router.get('/', verifyToken, (req, res) => {
  res.json({ success: true, message: 'Protected resource' });
});
router.post('/shrinkUrl', verifyToken, linkController.shrinkUrl);
router.post('/addLink', verifyToken, linkController.addLink);
router.get('/getLinkByUrl', verifyToken, linkController.getLinkByUrl);
router.get('/getLinkByShortUrl', verifyToken,
linkController.getLinkByShortUrl);

export default router;
```

Server initialization file, provide all necessary actions for application work. This file set up all parser features for http requests as well as set up the CORS rules to provide security management of data between parts of app and prevent connection from unknow sources.

File use SequelizeDB feature imported from “Sequelize” config file. Dotenv imports app credentials. Express - setup the rules of parsing and main settings of express app.

Listing 8 describes the server size code of “server file”.

File “server.ts”

```
import express from 'express';
import router from './routes/app-routes';
import { SequelizeizeDB } from './db/sequelizeize.db';
import dotenv from 'dotenv';

dotenv.config();
const _frontend_port = process.env.FRONTEND_PORT;
const _frontend_host = process.env.FRONTEND_HOST;
const db = new SequelizeizeDB();
const cors = require('cors');
const path = require('path');
const multer = require('multer');
const upload = multer();
const app = express();

const corsOptions = {
  origin: `http://${_frontend_host}:${_frontend_port}`,
  credentials: true,
  methods: ['GET', 'POST'], allowedHeaders: ['Content-Type',
'Authorization']
};

app.use(cors(corsOptions));
app.options('*', cors(corsOptions));
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(upload.none());
app.use(express.static(path.join(__dirname, 'public')));

app.use('/', router);

db.openConnection().then(() => {
  const PORT = parseInt(process.env.APP_PORT || '8080');

  app.listen(PORT, () => {
    console.log(`The server is running on port ${PORT}`);
  });
  db.createTables().then(() => {
    console.log('Model for db created');
  });
});
```

Authentication page file, responsible for authentication user with two types of security, app auth or google auth. This two options both have token system which will be provided after auth code 200 “OK”.

This file use many react features like Link for creation of app move by urls, useHistory for redirecting, useLocation, etc. It also has google package feature for wrapping react component with googleOAuth, and the logic of google authentication with GoogleLogin. For token which return by google we use jwt decode feature for parsing the info from token, to continue our verification.

Listing 9 describes the auth page of “Auth Page” react component.

Listing 9

File “AuthPage.js”

```
import React, { useState, useEffect } from "react";
import { Link, useHistory, useLocation } from "react-router-dom";
import { GoogleOAuthProvider } from "@react-oauth/google";
import { GoogleLogin } from "@react-oauth/google";
import isGoogleTokenValid from "../api/googleTokenCheck";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import { faGoogle } from "@fortawesome/free-brands-svg-icons";
import { jwtDecode as jwt_decode } from "jwt-decode";

const _host = process.env.REACT_APP_BACKEND_HOST;
const _port = process.env.REACT_APP_BACKEND_PORT;
const _google_client_id = process.env.REACT_APP_GOOGLE_CLIENT_ID;

const AuthPage = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState("");
  const [success, setSuccess] = useState("");
  const history = useHistory();
  const location = useLocation();

  useEffect(() => {...}, [location]);

  const handleLogin = (e) => {...};
```

```

const onSuccessGoogleLogin = async (res) => {...};
const result = await isGoogleTokenValid(doc.token);
  if (!result) {
    setError("Google token is not valid");
  } else {
    const formData = new FormData();
    formData.append("username", doc.username);
    formData.append("email", doc.email);
    fetch(`http://${_host}:${_port}/auth/googleuser`, {...});
  }
} catch (error) {
  console.error(
    "A network error occurred when trying to fetch resource:",
    error
  );
}
};
const onFailureGoogleLogin = async (res) => {...};
return (...);

export default AuthPage;

```

Login page inherit logic of Register page file, it has create user account in a system with two types of authentication. This account will have unique token to connect with the system each time the user log in by his credentials.

The file have similar packages of login page but i have different http request witch will use different route of backend part of app.

Listing 10 describes registration page react component

Listing 10

File “RegisterPage.js”

```

import React, { useState } from "react";
import { Link } from "react-router-dom";
import { GoogleOAuthProvider } from "@react-oauth/google";
import { GoogleLogin } from "@react-oauth/google";
import { useHistory } from "react-router-dom";
import isGoogleTokenValid from "../api/googleTokenCheck";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import { faGoogle } from "@fortawesome/free-brands-svg-icons";
import { jwtDecode as jwt_decode } from "jwt-decode";
import { faExclamationTriangle } from "@fortawesome/free-solid-svg-icons";

```

```

const _google_client_id = process.env.REACT_APP_GOOGLE_CLIENT_ID;
const _host = process.env.REACT_APP_BACKEND_HOST;
const _port = process.env.REACT_APP_BACKEND_PORT;

const RegisterPage = ({ ClientId }) => {
  const [username, setUsername] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const [error, setError] = useState("");
  const history = useHistory();

  const handleRegistration = async (event) => {...};
  const onSuccessGoogleLogin = async (res) => {...};
  const onFailureGoogleLogin = async (res) => {...};
  return {...};
};
export default RegisterPage;

```

Main page responsible for main software interface where the user can make some actions with the links, like send long links to api or browse the lists of already shrank links. Copy that links and use it or just add favorite and unfavorite links. The main page handles the user data with different buttons. User also can use different links to redirect himself by some interested pages. Main page nav bar will receive user info like “username” and “user icon” and display it on the upper right corner

Listing 11 describes main page react component

Listing 11

File “MainPage.js”

```

import React, { useEffect, useState } from "react";
import { useHistory } from "react-router-dom";
import NavBar from "../components/Navbar";
import Footer from "../components/Footer";
import LinkForm from "./LinkForm";
import RecentLinks from "./RecentLinks";

const _host = process.env.REACT_APP_BACKEND_HOST;
const _port = process.env.REACT_APP_BACKEND_PORT;

```



```

const MainPage = () => {
  const history = useHistory();

  const [urlArray, setUrlArray] = useState([]);
  const [shortUrlArray, setShortUrlArray] = useState([]);

  const handleLinkFormData = (newUrl, newShortUrl) => {...};

  setShortUrlArray((oldArray) => {...});

  useEffect(() => {
    const token = sessionStorage.getItem("token");
    const username = sessionStorage.getItem("username");
    if (!token && !username) {
      history.push("/login");
    }
  }, [history]);

  return (...);
}

export default MainPage;

```

Link form responsible for creation of user interface form component in which user can make actions related to links and api that can make short versions of the links. Contain copy method, which connected to inputs in that form and make copy function as the effect of the user click the button.

Listing 12 describes the link form component.

Listing 12

File "LinkForm.js"

```

import React, { useState } from "react";
import { FontAwesomeIcon } from "@fortawesome/react-fontawesome";
import { faCopy } from "@fortawesome/free-solid-svg-icons";
import { useRef } from "react";

const LinkForm = (props) => {
  const [link, setLink] = useState("");
  const [shortLink, setShortLink] = useState("");
  const inputRef = useRef();

  const handleLinkSubmit = (event) => {...}
  const handleCopy = () => {...};

```

```

    return (...);
};
export default LinkForm;

```

Navbar is a react component, can be integrated as part of each component page, his responsible for navigation between pages by using Link features and contain info about the application Company and User info.

Listing 13 describes the navbar react component.

Listing 13

File “Navbar.js”

```

import React from "react";
import { Link, useHistory } from "react-router-dom";
import userIcon from "../img/User-icon.png";
import logo from '../img/Logo.png';
const NavBar = (props) => {
  const history = useHistory();

  let userAvatar;
  const username = props.username ? props.username : "Anonymous";

  if (props.picUrl) {
    userAvatar = props.picUrl;
  } else {
    userAvatar = userIcon;
  }

  function handleLogout() {...}

  return (...);
};
export default NavBar;

```

“Not Found” page component refers to error type of components. User can see displayed “not found” page only by manually redirecting himself to some url which has not any prerendered content, in that way user will be informed that he is not on a right page.

This file does not use any of packages or features but only simple react.

Listing 14 describes “Not Found” component.

File “NotFound.js”

```
import React from "react";

const NotFoundPage = () => {
  return (...);
};

export default NotFoundPage;
```

Recent Links is a react component which display a list of lists combined with their short links already converted with API. As well as the Link Form component it has a handle copy function for each group of links to copy that for comfort user experience.

Listing 15 describes recent links component.

File “RecentLinks.js”

```
import React from "react";

const RecentLinks = ({ urlArray, shortUrlArray }) => {

  const handleCopy = (url) => {...};

  return (...);
};

export default RecentLinks;
```

App file is a main react configuration file, which hold all react components and provide woking frontend part of app. It has import all the components and use their logic. Also using BrowserRouter to switch between paths (some sort of navigation), router, route and switch features that cannot be without this main package. The architecture of that type of app should have all wrapped routes in switch to change the pages of app by user actions as well as imported components as the parameters in each route to return the content of some page to user.

Listing 16 describes app react file.

Listing 16

File "index.js"

```
import React from "react";
import ReactDOM from "react-dom";
import { BrowserRouter as Router, Route, Switch } from "react-router-dom";
import AuthPage from "../auth/AuthPage";
import MainPage from "../components/MainPage";
import NotFoundPage from "../components/NotFoundPage";
import RegisterPage from "../auth/RegistrationPage";
import "../sass/main.scss";

import { BrowserRouter } from "react-router-dom/cjs/react-router-dom.min";

const App = () => {
  return (
    <BrowserRouter>
      <Router>
        <Switch>
          <Route exact path="/" component={MainPage} />
          <Route exact path="/login" component={AuthPage} />
          <Route exact path="/register" component={RegisterPage} />
          <Route component={NotFoundPage} />
        </Switch>
      </Router>
    </BrowserRouter>
  );
};

ReactDOM.render(<App />, document.getElementById("root"));
```

4.3. User interface development.

The launch page of the web service is the "Mainpage.js" file, this is the main page that immediately checks the attempt to receive data from the application for the presence of a token confirming his identity by the subject who made the request. In the usual version, when the user has not yet created an account in the application, he is redirected to the authentication page, and we will start with it.

This page requires two lines of data from the user, e-mail name and account password.

Figure 1 shows the web service login page.

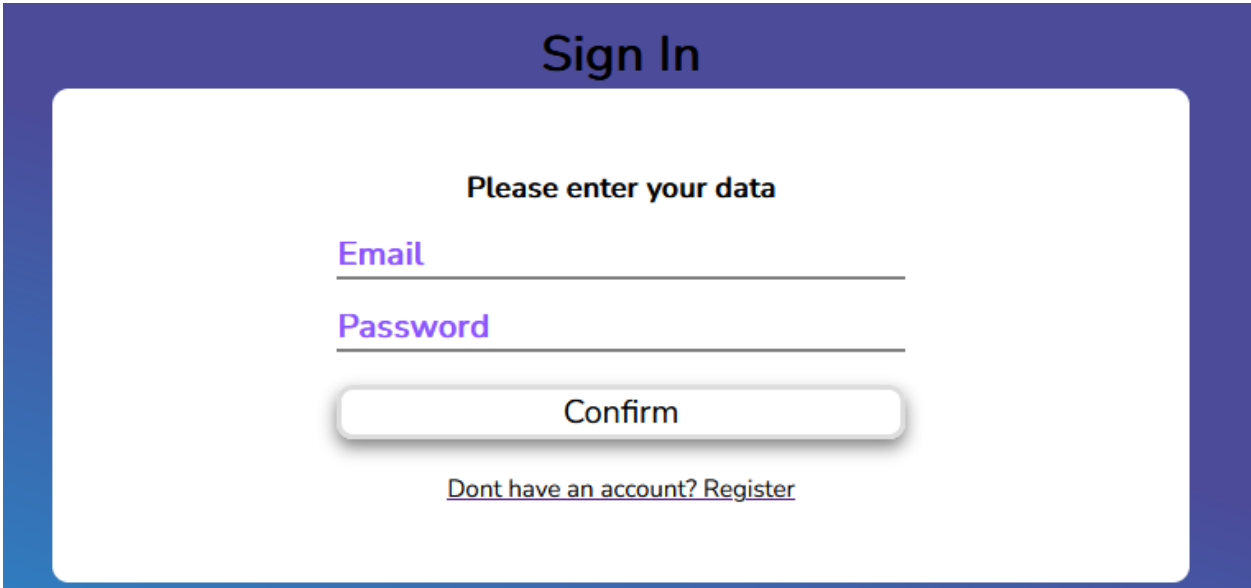


Fig. 4.1. “Login page”

As you enter data into the cells, the system will evaluate and change the color of the bottom lines to show the status of the data, whether it meets the required data entry rules or not. In addition, standard information about incorrect data entry will be displayed thanks to the browsers themselves.

At the moment of clicking a button in the interface form, the browser checks the data entered in the form for compliance with the conditions. If these conditions are not met, the form submission will be canceled. After that, the user will be prompted for information on how to correctly fill in the form fields. If these fields are successfully filled with data in the required form, the system will accept this information and take appropriate actions.

Figure 2 shows the web service login page with incorrect data fields.

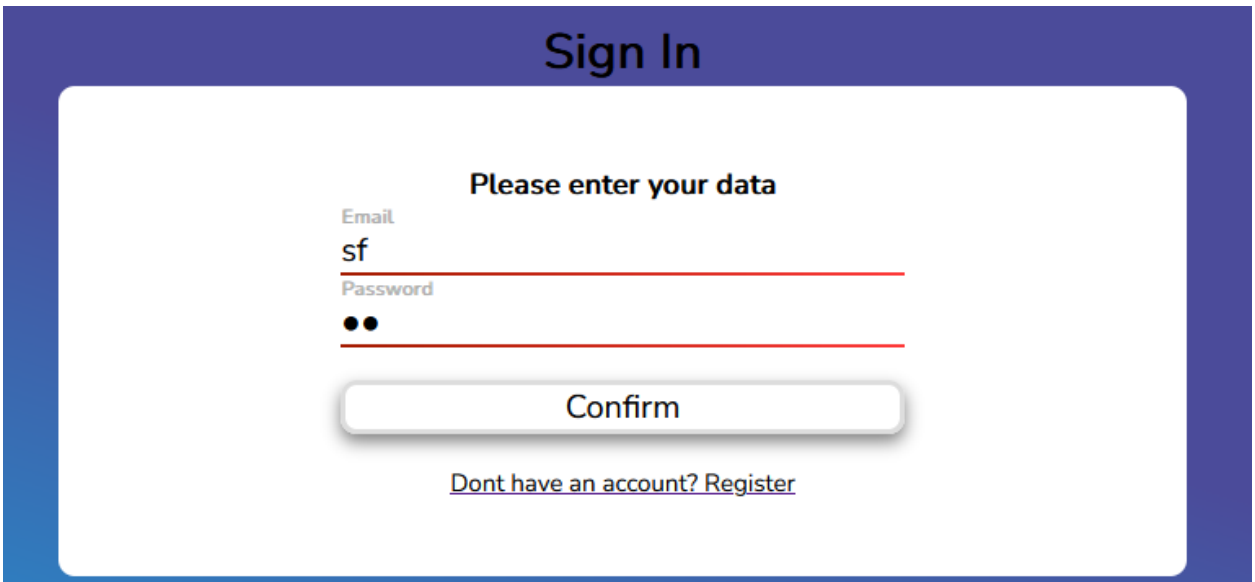


Fig. 4.2. "Error login page"

Entering data into the fields of the form in the correct form before sending the data itself by pressing the button will be highlighted in green by the lines under each field.

Figure 3 shows the login page to the web service with correctly filled form fields.

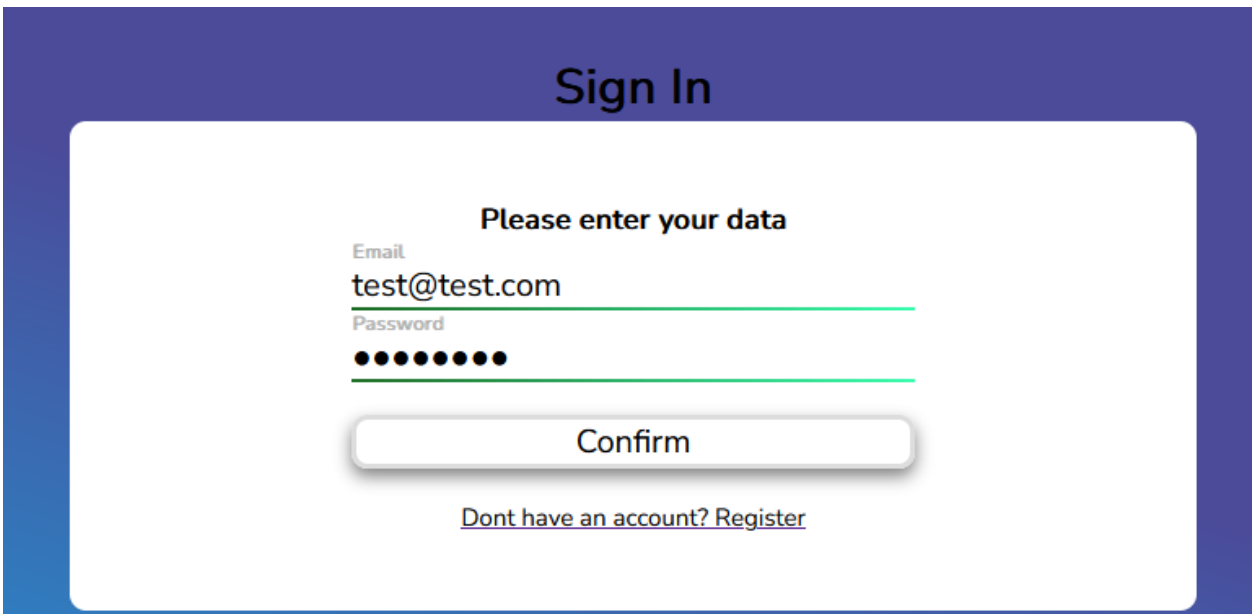


Fig. 4.3. "Error login page"

If the user does not have an account created in the corresponding service, he can quickly go to the account registration form by clicking on the link located under the button to send the "Confirm" form. After these actions, the user will be redirected to a similar page, but already intended for user registration.

The difference from the previous form is that it has an additional cell with the name "username", which requires the user to enter his nickname, which will be displayed on the main page of the web service. Also, one of the differences are the methods that will be performed when sending the form, this is the creation of a new account in the database, and not a check for the presence of this account.

Figure 4 shows the login page to the web service with correctly filled form fields.

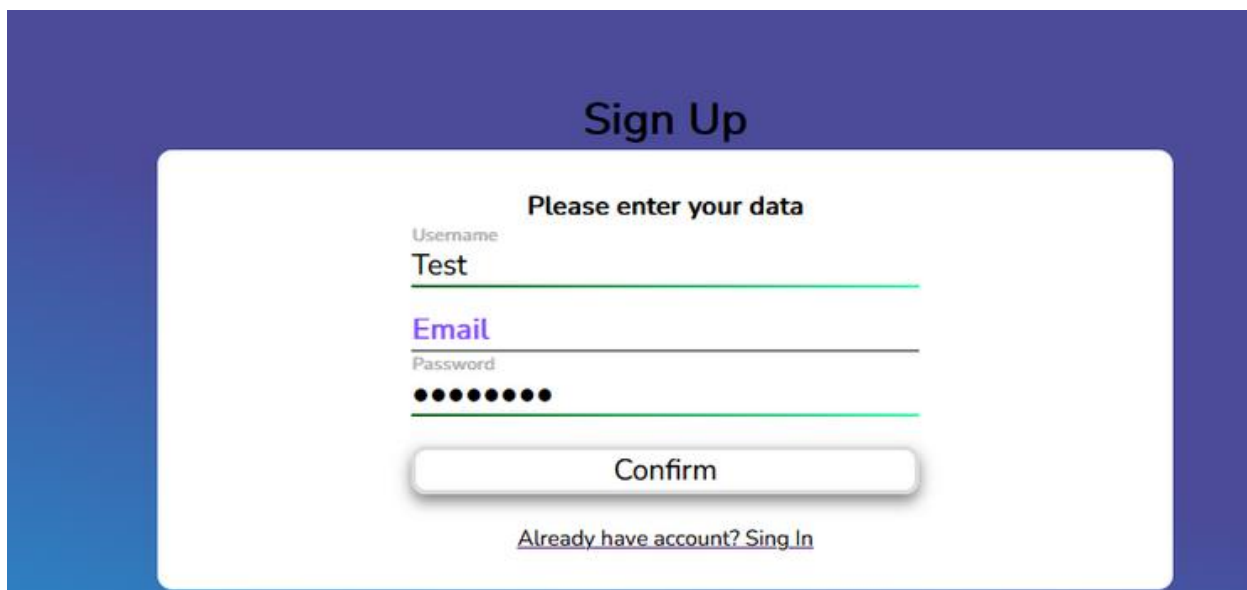
A screenshot of a web registration form titled "Sign Up". The form is centered on a white background with a blue border. It contains the following elements: a heading "Please enter your data", a "Username" field with the value "Test", an "Email" field, a "Password" field with masked characters (dots), a "Confirm" button, and a link "Already have account? Sing In".

Fig. 4.4. "Registration page"

One of the possible authentication options is also the use of Google OAuth functions, which provides the possibility of entering the application through the use of an already existing Google account, which in turn exchanges data with the service and receives the necessary user information and adds it to its system. The display name and image of the Google account will also be received and displayed later on the main page. Figure 5 shows the Google authentication button.

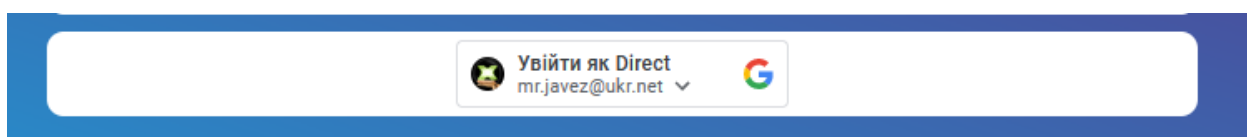


Fig. 4.5. "Google authentication button"

After pressing the button to log in using google authentication, a standard form will open for the user to choose the account with which he wants to use our application.

Figure 6 shows the Google account selection form.

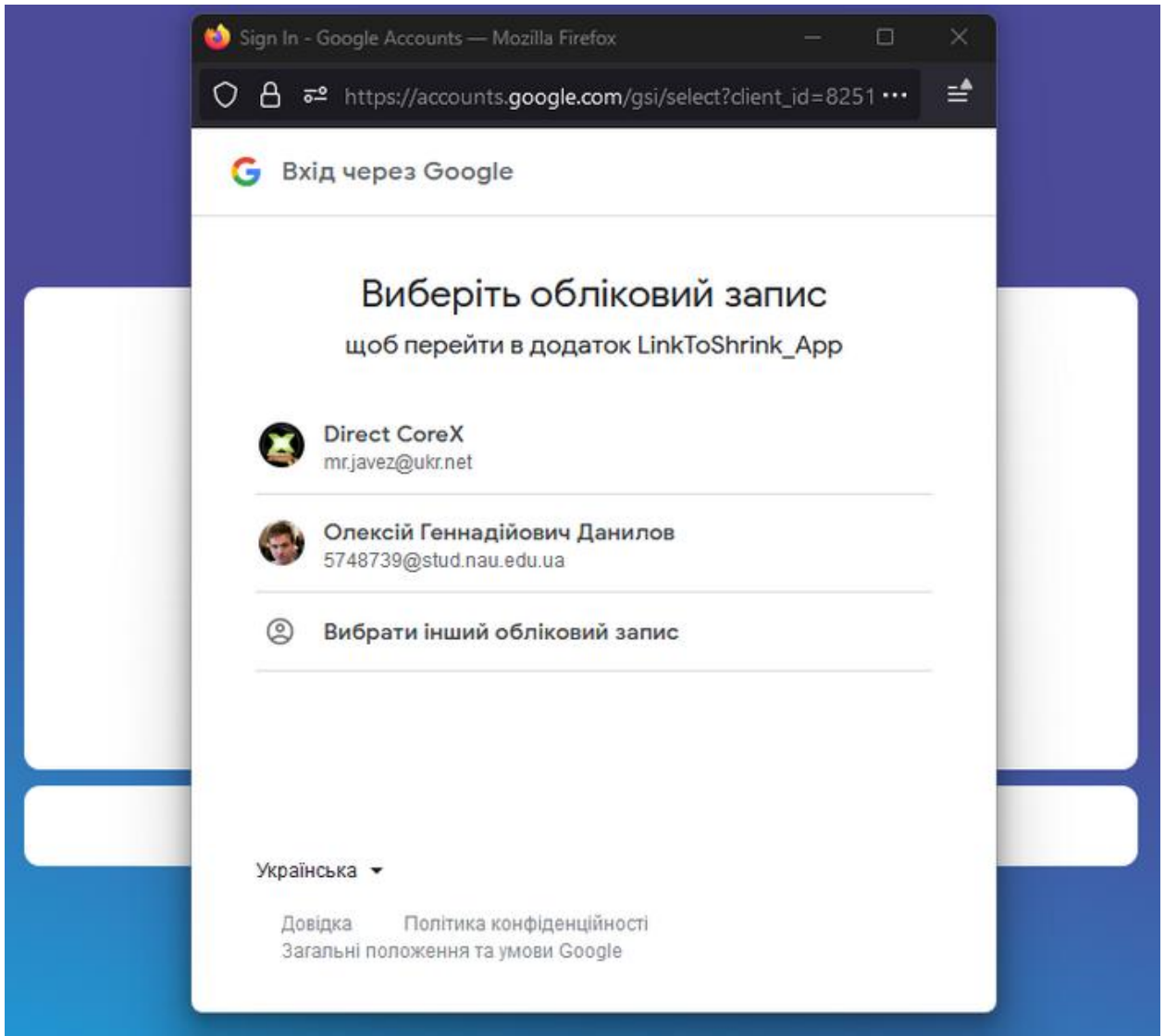


Fig. 4.6. “Google authentication form”

Handling of major errors that may occur while working with the system at times when its parts are under maintenance or for some reason an important error in the system has been discovered also takes place. Therefore, in some moments of system errors, for the user to understand the root of the problem, it was chosen to create additional interface components that display these errors, in their usual form, they will look like a drop-down block with information in red or green color if the operation was successful. However, unlike the information shown in the picture, it will have more understandable information for the user.

Figure 7 shows a form with an error block when performing some operation.

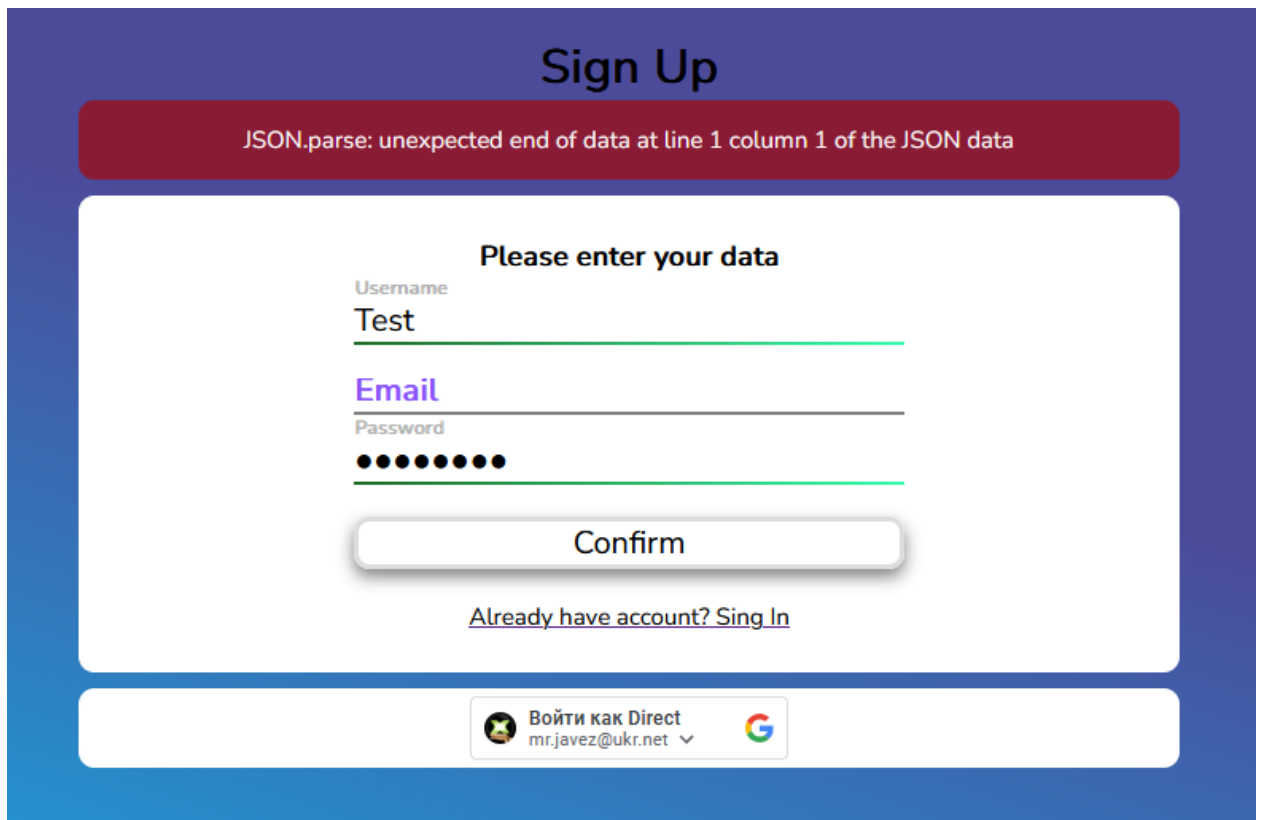


Fig. 4.7. "Google authentication button"

After successful registration and authorization in the system, the user will be redirected to the main page of the system. This page displays basic user data, buttons that can be used to go to other resources, etc. The main form of the system is shortening of links sent by the user. In order to perform operations, the user needs to independently select a link or a package of links and send these links to the system through the field for entering the link and the submit button of the form.

In addition, by following the links by pressing the buttons, the user can get various pages with information for familiarization. For example, the button called "about" will open a page with a description of the system and its capabilities. The "Company" button provides information about the company that owns this service. The "Support" button helps the company to improve the system.

Figure 8 shows the main page of the application.

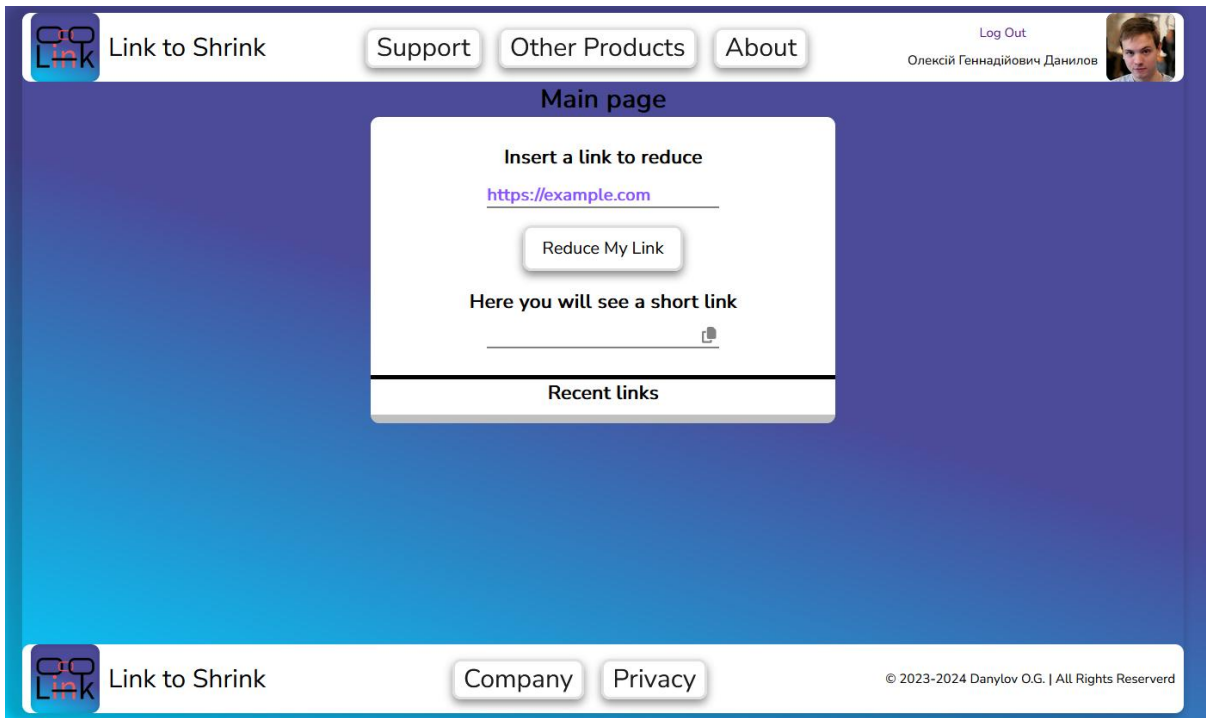


Fig. 4.8. “Main page”

After sending a certain number of links, the answer together with the sent link will be stored and displayed under the form for sending data, the number of storage of the last links was programmatically limited to no more than 10 so that the user does not make too large lists of links that he may not need. Next to these links will be a copy button that will allow you to quickly copy already converted links.

Figure 9 shows the main page of the application with recently used links.

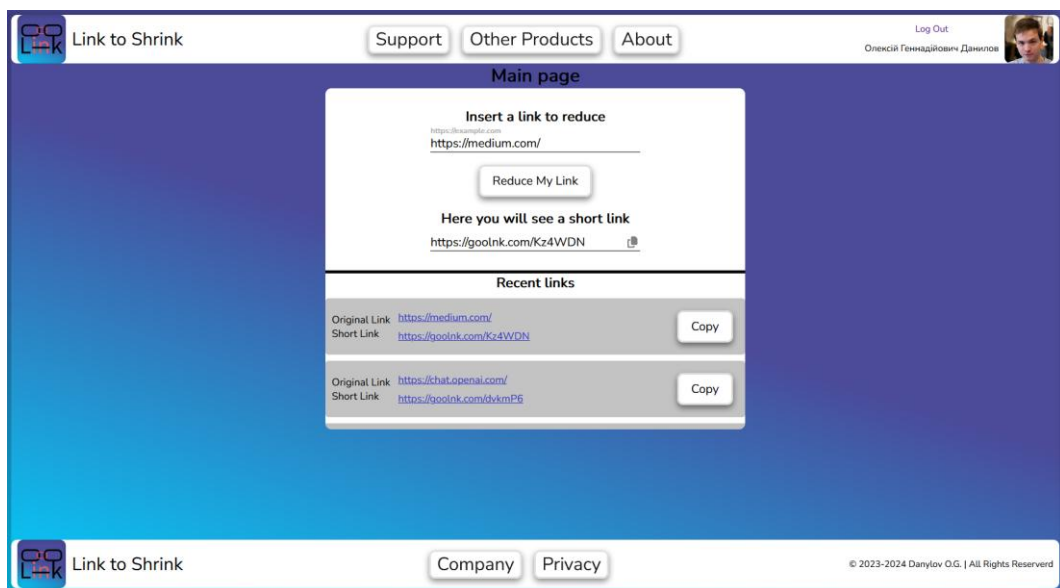


Fig. 4.9. “Main page with recent links”

4.4. Results of web link reduction system testing.

Having constructed a prototype of the software, preliminary testing of separate parts of the software was carried out, functions, components, units of software modules and the interaction of units with each other were involved.

In general, only the part with a complex authentication architecture was difficult in testing the system due to the creation of user tokens based on receiving data about his registration in the system and further saving this data to keep a certain user in the system. In addition, the ORM Sequelize configuration file was difficult to test, which is why the creation of tests around such a system was hidden due to security issues.

An example of testing the backend part of a web service. Testing includes almost all software files that can be tested. Some of the software components have little coverage due to the low need for testing software modules that perform a light load on the system.

Figure 4.10 presents the results of preliminary testing of the back-end part of the system prototype obtained from frameworks for software testing.

File	% Stmts	% Branch	% Funcs	% Lines
All files	63.7	35.41	68.88	61.94
api	100	100	100	100
shrink-api.ts	100	100	100	100
controllers	72.36	37.5	100	69.56
...troller.ts	72.36	37.5	100	69.56
db	43.24	50	33.33	43.24
redis.db.ts	100	50	100	100
...lize.db.ts	30	50	20	30
db/models	100	100	100	100
models.ts	100	100	100	100
...leware/auth	86.36	33.33	100	85
auth.ts	86.36	33.33	100	85
routes	100	100	100	100
app-routes.ts	100	100	100	100
services	44.31	0	44.44	43.02
service.ts	44.31	0	44.44	43.02

Fig. 4.10. "Backend test results"

When testing front-end parts, it is necessary to take into account the fact that separate files with functions that perform business logic on the client side, as well as user interface components that can also include some simpler functions but perform the necessary actions to change display information, can be tested in the application interface. Program files are logically sorted into folders that are responsible for individual functions in the system.

In this way, it is possible to obtain data that during testing, the software module that was difficult to verify was the testing of registration and authentication forms, because they, in addition to the usual logic and requests to the back-end part, perform authentication logic using Google and also send requests to it to receive valid tokens for further their confirmation.

Figure 4.11 presents the results of preliminary testing of the front-end part of the system prototype obtained from frameworks for software testing.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	58.85	35.41	60	58.85	
api	80	100	100	80	
..TokenCheck.js	80	100	100	80	15-16
auth	52.13	32.14	52	52.13	
AuthPage.js	50.81	31.25	57.14	50.81	..06,114-118,199
..rationPage.js	53.57	33.33	45.45	53.57	..01,109-113,218
components	67.69	33.33	68.42	67.69	
Footer.js	100	100	100	100	
LinkForm.js	77.27	50	71.42	77.27	26-28,37,42
MainPage.js	52	25	40	52	18-35
Navbar.js	60	50	50	60	12,18-21
RecentLinks.js	100	100	100	100	
					Test

Fig. 4.11. "Frontend test results"

We have two minimum condition during testing of the front-end part. First is the processing of the most important modules that are responsible for the security of all web service operations. Secondly is the processing of the minimum half 50% of the entire system with tests coverage both separate and integration tests (for the interaction of these modules).

Conclusion

In this section, the final part of the work on software development using the methodology of code generation by artificial intelligence was considered. A detailed description of the design and development aspects of the software prototype as a whole was made.

Also considered what is the software architecture and the requirements for it when choosing an architecture according to the system to be developed with presentation of the main software modules of the system.

Made description of the final requirements for the appearance of the system interface, including the most important pages that will be used by users.

After constructing the prototype of the link shortening service, software testing of the entire system, i.e., two parts of the service, was also created.

Separately created unit and integration testing of the front-end part, its react components and functions. The unit and integration testing of the front-end part, its service parts and function APIs have been created.

CONCLUSIONS

In conclusion, the integration of artificial intelligence (AI) techniques, including IntelliSense, Code Formatter, ChatGPT, and GitHub Copilot, in development software systems offers tremendous potential for improving the efficiency and effectiveness of the code creation process. The combination of these AI-driven tools provides benefits in code comprehension, automated refactoring, and documentation generation.

By leveraging IntelliSense, development can gain a deeper understanding of complex code structures, identify relationships between software components, and obtain contextual suggestions for code completion. This enhances code comprehension and reduces the manual effort required for analyzing and understanding intricate codebases.

Code Formatter automates refactoring tasks, making legacy codebases more maintainable and aligned with coding best practices. It improves code readability, structure, and modularity, thereby facilitating the code creation process and enabling developers to work with cleaner and more organized code.

ChatGPT proves valuable in generating comprehensive and up-to-date documentation during code generation. It assists in documenting code logic, identifying system dependencies, and providing explanations and summaries. This AI-powered documentation generation saves time and effort, ensuring that crucial information about the software system is captured accurately.

GitHub Copilot, with its intelligent code generation capabilities, expedites the development process by suggesting relevant code snippets, functions, and patterns based on observed context. This AI-powered tool assists during development in identifying critical functionality and accelerates the understanding of complex code structures within the software system.

However, it is important to consider the challenges associated with AI-driven code generation. Ensuring the availability of accurate and representative training data, addressing biases in AI models, and ensuring the trustworthiness and reliability of AI-generated code and documentation are critical considerations.


Future directions for research and development in AI-driven code generation include improving AI models' contextual understanding, addressing security and privacy concerns, and fostering effective collaboration between AI and human expertise.


In summary, the integration of AI techniques in module development of software systems has the potential to revolutionize the field, improving code comprehension, refactoring efficiency, and documentation generation. It enables code generation to navigate complex codebases more effectively, enhance maintainability, and gain deeper insights into software systems, ultimately leading to more efficient and informed software development and maintenance processes.

LIST OF REFERENCES

1. Beiqi Zhang, Peng Liang, Xiyu Zhou, Aakash Ahmad, «*Practices and Challenges of Using Github Copilot: An Empirical Study*», March 2023, [Electronic resource] - Mode of access: https://www.researchgate.net/publication/369266175_Practices_and_Challenges_of_Using_GitHub_Copilot_An_Empirical_Study
2. Mateusz Jaworski, Dariusz Piotrkowski, «*Studios of software developers experience using the Github Copilot Tool in the software development process*», January 2023, [Electronic resource] - Mode of access: https://www.researchgate.net/publication/367088653_Study_of_software_developers%27_experience_using_the_Github_Copilot_Tool_in_the_software_development_process
3. Burak Yetiştiren, Eray Tüzün, Işık Özsoy, «*Assessing the Quality of GitHub Copilot's Code Generation*», November 2022, посилання [Electronic resource] - Mode of access: https://www.researchgate.net/publication/363267006_Assessing_the_Quality_of_GitHub_Copilot%27s_Code_Generation
4. Meiyappan Nagappan, N. Asokan, «*Is GitHub's Copilot as Bad As Humans at Introducing Vulnerabilities in Code*», April 2022, [Electronic resource] - Mode of access: https://www.researchgate.net/publication/359890768_Is_GitHub%27s_Copilot_as_Bad_As_Humans_at_Introducing_Vulnerabilities_in_Code
5. Tosh Velaga, «*Code generation Resources for ChatGPT*» Dec 12, 2022 [Electronic resource] - Mode of access: <https://medium.com/@toshvelaga/reverse-engineering-resources-for-chatgpt-bd69155b2423>
6. Aleksandar Milenkoski, «*Problems Chatgpt Can Solve for Code Generation and Malware Analysis*» December 2022, [Electronic resource] - Mode of access: <https://www.sentinelone.com/labs/11-problems-chatgpt-can-solve-for-reverse-engineers-and-malware-analysts/>
7. Рябокінь Ю.М., «Інженерія програмного забезпечення» НАУ, 2015 –184с.;
8. Крамар Ю.М., Рябокінь Ю.М., Кузнецов С.В., Марченко О.І., Програмне забезпечення автоматизованих систем, НАУ 2006 –38с.;
9. Радішевський М.Ф., Рябокінь Ю.М., Оцінювання об'єктно-орієнтованих програмних систем на етапі проектування, НАУ 2009 – 6с.;
10. Сидоров М.О., Вступ до інженерії програмного забезпечення, НАУ 2010 –112с.;

ADDITIONAL PICTURES

 Help me i need idea how can i improve created backend API app by changing the type of db MySQL to PostgreSQL, by using Sequelize. Project technology stack is: Typescript, Node.js

 Migrating your backend API app from MySQL to PostgreSQL using Sequelize is a good idea, and it can be done relatively smoothly. Here are the steps to help you improve your application:

1. **Backup Data:** Before making any changes, make sure to create a backup of your MySQL database. This is crucial to ensure you don't lose any data during the migration process.
2. **Install Sequelize and Sequelize CLI:**

```
npm install sequelize sequelize-cli pg pg-hstore
```
3. **Create a PostgreSQL Database:** Set up a PostgreSQL database if you haven't already. You will need the connection details (e.g., host, port, username, password, database name) for Sequelize configuration.
4. **Create Sequelize Models:** Sequelize allows you to define models that represent your database tables. You'll need to create new Sequelize models that match your existing MySQL tables. You can use the `sequelize-auto` tool to generate models based on your existing MySQL database schema.


```
arduino
```

```
npx sequelize-auto -o "./models" -d your_old_mysql_db -h mysql_host -u mysql_username -P mysql_password
```
5. **Modify Sequelize Config:** Update your Sequelize configuration to point to the PostgreSQL database. The configuration is typically found in a file like `config/config.json`. Replace the MySQL configuration with PostgreSQL configuration.


```
json
```

```
{
  "development": {
    "username": "postgres",
    "password": "your_password",
    "database": "your_postgres_db",
    "host": "localhost",
    "dialect": "postgres"
  }
}
```
6. **Migrate Data:** Use Sequelize migrations to transfer the data from MySQL to PostgreSQL. Create a migration script that reads data from your MySQL database and inserts it into your PostgreSQL database.